

SOFTWARE DEVELOPMENT ESTIMATING HANDBOOK PHASE ONE



**CHERI CUMMINGS
MICHAEL GALLO
PAMELA JOHNSON
BARBARA MARSH-JONES
JILL von KUEGELGEN**

February 1998

Table of Contents

| SECTION | PAGE |
|--|------------|
| List of Tables..... | vi |
| List of Figures..... | viii |
| Executive Summary..... | ix |
| Acknowledgments..... | xx |
| 1.0 Introduction..... | 1-1 |
| 1.1 Handbook Introduction..... | 1-1 |
| 1.2 Handbook Overview..... | 1-2 |
| 2.0 Defining the Problem..... | 2-1 |
| 3.0 Software Database..... | 3-1 |
| 3.1 Introduction..... | 3-1 |
| 3.2 Ground Rules and Assumptions..... | 3-2 |
| 3.2.1 Assumptions for Sizing..... | 3-2 |
| 3.2.2 Assumptions for Scope and Distribution of Effort..... | 3-4 |
| 3.2.3 Other Assumptions..... | 3-5 |
| 3.3 Data Field Definitions..... | 3-5 |
| 3.4 Raw Data..... | 3-16 |
| 3.4.1 MITRE Non-Ada Database..... | 3-16 |
| 3.4.2 MITRE Ada Database..... | 3-18 |
| 3.4.3 Air Force Space and Missile Systems Center (SMC) Software Database..... | 3-19 |
| 3.4.4 NASA Software Engineering Laboratory (SEL) Database... | 3-21 |
| 3.4.5 Navy Internal Data..... | 3-23 |
| 3.4.6 Silver SASET Validation Database..... | 3-24 |
| 3.4.7 REVIC Recalibration Database..... | 3-25 |
| 3.4.8 IITRI Database..... | 3-25 |
| 3.5 Results..... | 3-26 |
| 3.6 Conclusions..... | 3-28 |
| 3.7 Future Efforts..... | 3-28 |
| 4.0 Effort Analysis: Significant Drivers..... | 4-1 |
| 4.1 Introduction..... | 4-1 |
| 4.2 Raw Data..... | 4-1 |
| 4.3 Methodology and Results..... | 4-2 |
| 4.3.1 Level One..... | 4-2 |
| 4.3.1.1 Mission..... | 4-2 |
| 4.3.1.2 Counting Convention..... | 4-6 |
| 4.3.1.3 Language..... | 4-7 |
| 4.3.1.4 Phasing..... | 4-9 |
| 4.3.2 Level Two..... | 4-11 |
| 4.3.3 Level Three (Program) and Level Four (CSCI)..... | 4-12 |
| 4.4 Conclusions..... | 4-20 |

| | |
|--|------------|
| 4.5 Weaknesses..... | 4-21 |
| 4.6 Future Efforts..... | 4-21 |
| 5.0 Effort Analysis: Normalized Regressions..... | 5-1 |
| 5.1 Introduction..... | 5-1 |
| 5.2 Review of the NCCA Normalized Software Effort Database..... | 5-1 |
| 5.3 Partitioning the Data..... | 5-4 |
| 5.4 Analytical Approach..... | 5-4 |
| 5.5 Regression Results..... | 5-7 |
| 5.5.1 Traditional Regressions..... | 5-8 |
| 5.5.1.1 Round One Eliminations..... | 5-9 |
| 5.5.1.2 Round Two Eliminations..... | 5-10 |
| 5.5.1.3 Round Three Eliminations..... | 5-10 |
| 5.5.2 Non-Traditional Regressions: Set One..... | 5-15 |
| 5.5.3 Non-Traditional Regressions: Set Two..... | 5-16 |
| 5.5.4 Revised Traditional Regressions..... | 5-18 |
| 5.6 Evaluation of Program-Level versus CSCI-Level Regressions..... | 5-19 |
| 5.7 Recommendations..... | 5-24 |
| 5.8 Conclusions..... | 5-26 |
| 5.9 Future Efforts..... | 5-28 |
| 6.0 Effort Analysis: Non-Normalized Productivity Factors..... | 6-1 |
| 6.1 Introduction..... | 6-1 |
| 6.2 Data..... | 6-1 |
| 6.3 Methodology and Results..... | 6-2 |
| 6.3.1 MIS Programs..... | 6-2 |
| 6.3.2 Weapon System Programs - Primarily Assembly..... | 6-4 |
| 6.3.3 Weapon System Programs - 100 Percent Assembly..... | 6-5 |
| 6.3.4 Physical Code Counting Convention..... | 6-6 |
| 6.3.5 Unknown Code Counting Convention..... | 6-7 |
| 6.3.6 Logical Code Counting Convention - Phasing Unknown..... | 6-7 |
| 6.4 Recommendations..... | 6-8 |
| 6.5 Conclusions..... | 6-9 |
| 6.5.1 Strengths..... | 6-9 |
| 6.5.2 Weaknesses..... | 6-9 |
| 7.0 Effort Analysis: Overall Process..... | 7-1 |
| 8.0 Schedule Analysis..... | 8-1 |
| 8.1 Introduction..... | 8-1 |
| 8.2 NCCA Schedule Databases..... | 8-1 |
| 8.2.1 Ground Rules and Assumptions..... | 8-1 |
| 8.2.2 Raw Schedule Database..... | 8-2 |
| 8.2.3 NCCA Normalized Schedule Database..... | 8-2 |
| 8.3 Methodology and Results..... | 8-4 |
| 8.3.1 Approach One..... | 8-4 |
| 8.3.2 Approach Two..... | 8-5 |
| 8.4 Recommendations..... | 8-7 |
| 8.5 Conclusions..... | 8-8 |
| 8.6 Future Efforts..... | 8-9 |

| | |
|---|-------|
| 9.0 Labor Rate Analysis | 9-1 |
| 9.1 Introduction | 9-1 |
| 9.2 NCCA Labor Rate Databases | 9-1 |
| 9.2.1 Ground Rules and Assumptions | 9-1 |
| 9.2.2 Data Sources | 9-2 |
| 9.2.3 Raw Labor Rate Database | 9-5 |
| 9.2.4 NCCA Normalized Labor Rate Database | 9-6 |
| 9.3 Methodology and Results | 9-7 |
| 9.3.1 Average Labor Rate Analysis | 9-8 |
| 9.3.2 Regression Analysis | 9-9 |
| 9.3.2.1 Regression Analysis Set One | 9-10 |
| 9.3.2.2 Regression Analysis Set Two | 9-10 |
| 9.4 Recommendations | 9-11 |
| 9.5 Conclusions | 9-13 |
| 9.6 Additional Considerations | 9-13 |
| 9.7 Future Efforts | 9-13 |
| 10.0 Risk Analysis | 10-1 |
| 10.1 Introduction | 10-1 |
| 10.2 NCCA Risk Analysis Databases | 10-1 |
| 10.2.1 Ground Rules and Assumptions | 10-1 |
| 10.2.2 Raw Risk Analysis Database | 10-2 |
| 10.2.3 NCCA Normalized Risk Analysis Database | 10-2 |
| 10.3 SLOC Growth Methodology, Results and Conclusions | 10-3 |
| 10.3.1 Approach One | 10-3 |
| 10.3.1.1 Approach One Methodology and Results | 10-3 |
| 10.3.1.2 Approach One Conclusions | 10-4 |
| 10.3.2 Approach Two | 10-5 |
| 10.3.2.1 Approach Two Methodology and Results | 10-5 |
| 10.3.2.2 Approach Two Conclusions | 10-6 |
| 10.3.3 Approach Three | 10-7 |
| 10.3.3.1 Approach Three Methodology and Results | 10-7 |
| 10.3.3.2 Approach Three Conclusions | 10-9 |
| 10.3.4 Recommended Approach | 10-9 |
| 10.4 Code Condition Change Methodology, Results and Conclusions | 10-10 |
| 10.4.1 Approach One | 10-10 |
| 10.4.1.1 Mean Percentage | 10-10 |
| 10.4.1.2 Median Percentage | 10-11 |
| 10.4.1.3 Mean Percentage Points | 10-11 |
| 10.4.1.4 Median Percentage Points | 10-12 |
| 10.4.2 Approach Two | 10-13 |
| 10.4.3 Recommended Approach for Code Condition Change | 10-14 |
| 10.5 Overall Recommended Approach | 10-15 |
| 10.5.1 Example One | 10-16 |
| 10.5.2 Example Two | 10-18 |
| 10.9 Future Efforts | 10-19 |
| 11.0 Conclusions | 11-1 |

Appendix A – Defining the Problem

| | |
|---|------|
| NCCA Software Program Definition Form..... | A-1 |
| NCCA Software Program Definition Form, Data Dictionary..... | A-4 |
| NCCA Historical Software Data Request Form..... | A-9 |
| NCCA Historical Software Data Request Form, Data Dictionary..... | A-13 |
| NCCA Historical Software Data Request Form's Mapping Procedures..... | A-20 |

Appendix B – Software Database

| | |
|---|------|
| NCCA Raw Software Effort Database Key..... | B-1 |
| Navy Internal Data Sources..... | B-11 |
| NCCA Raw Software Effort Database: Duplicate Data Points..... | B-15 |

Appendix C – Effort Analysis: Significant Drivers

| | |
|---|------|
| Statistical Measures, Methods, and Procedures..... | C-1 |
| Results: Section 4 - Effort Analysis: Significant Drivers..... | C-9 |
| Mann-Whitney U Test Results: Section 4 - Effort Analysis: Significant Drivers..... | C-13 |

Appendix D – Effort Analysis: Normalized Regressions

| | |
|--|-------|
| Program-Level Regressions (Traditional/Non-Traditional/Revised Traditional - One Efactor)..... | D-1 |
| CSCI-Level Regressions (Traditional/Non-Traditional/Revised Traditional - One Efactor)..... | D-10 |
| Program-Level Regressions (Traditional - Two Efactors)..... | D-19 |
| CSCI-Level Regressions (Traditional - Two Efactors)..... | D-25 |
| Final Program-Level Regressions: Backup Data Traditional..... | D-31 |
| Non-Traditional..... | D-40 |
| Revised Traditional..... | D-47 |
| Final CSCI-Level Regressions: Backup Data Traditional..... | D-57 |
| Non-Traditional..... | D-69 |
| Revised Traditional..... | D-78 |
| Percent New Tradeoff (Program-Level)..... | D-91 |
| Comparison of Traditional, Non-Traditional and Revised Traditional Regressions (Program- versus CSCI-Levels)..... | D-99 |
| Comparison of Traditional, Non-Traditional and Revised Traditional Regressions (Program- and CSCI-Levels)..... | D-102 |

Appendix E – Schedule Analysis

| | |
|---|------|
| NCCA Raw Schedule Database..... | E-1 |
| NCCA Normalized Schedule Database..... | E-2 |
| Non-Parametric Test Results..... | E-3 |
| Statistical Results of the NCCA Normalized Schedule Database..... | E-10 |

| | |
|---|------|
| Non-Significant Schedule Regressions..... | E-15 |
| Schedule Regression Analysis..... | E-16 |
| Equation [8-1]..... | E-16 |
| Equation [8-2]..... | E-22 |
| Equation [8-3] | E-29 |
| Equation [8-4] | E-36 |
| Equation [8-5]..... | E-43 |
| Equation [8-6]..... | E-50 |

Appendix F – Labor Rate Analysis

| | |
|--|------|
| NCCA Labor Rate Database Key..... | F-1 |
| FPRA and Contractor-Specific Labor Rate Application..... | F-2 |
| Labor Rate Analysis: Factor and Regression Analyses..... | F-6 |
| Factor Analyses..... | F-6 |
| Regression Analyses..... | F-12 |
| Wilcoxon Two-Sample Test Results..... | F-19 |
| Average Burden Rates..... | F-24 |

Appendix G – Risk Analysis

| | |
|---|------|
| SLOC Growth CSCI-Level Analysis and Data..... | G-1 |
| AIS/MIS Data Points..... | G-3 |
| NCCA Raw SLOC Growth Database..... | G-4 |
| NCCA Normalized SLOC Growth Database..... | G-5 |
| Non-Parametric Test Results..... | G-6 |
| Regression Analyses (SLOC Growth – Approach Two)..... | G-9 |
| Regression Analyses (SLOC Growth – Approach Three)..... | G-45 |

List of Tables

| | | |
|------|--|------|
| 1 | Evolution of the NCCA Normalized Software Effort Database..... | xii |
| 2 | NCCA Normalized Software Effort Database and Source Databases..... | xiii |
| 3-1 | Allocation of New HOL and New Assembly SLOC..... | 3-3 |
| 3-2 | NCCA Raw Database Summary..... | 3-26 |
| 3-3 | NCCA Raw Database Data Sources..... | 3-27 |
| 3-4 | Summary of the NCCA Raw Database by Programming Language..... | 3-27 |
| 3-5 | Summary of the NCCA Raw Database by Mission Area..... | 3-27 |
| 3-6 | Summary of the NCCA Raw Database by Platform..... | 3-27 |
| 4-1 | MIS and Weapon System Data Sets..... | 4-3 |
| 4-2 | ESLOC Methods (Strengths and Weaknesses) | 4-5 |
| 4-3 | Level One Statistical Results (Mission) | 4-6 |
| 4-4 | Physical and Logical Data Sets..... | 4-7 |
| 4-5 | Level One Statistical Results (Counting Convention) | 4-7 |
| 4-6 | HOL and Assembly Data Sets..... | 4-8 |
| 4-7 | Level One Statistical Results (Language) | 4-8 |
| 4-8 | Normalized and Partially Normalized Data Sets..... | 4-10 |
| 4-9 | Level One Statistical Results (Phasing) | 4-10 |
| 4-10 | Program, CSCI, and 1CSCI Data Sets..... | 4-11 |
| 4-11 | Level Two Statistical Results..... | 4-11 |
| 4-12 | Level Three and Level Four Data Sets..... | 4-14 |
| 4-13 | Level Three and Level Four Statistical Results (Code Condition) | 4-15 |
| 4-14 | Level Three and Level Four Statistical Results (Platform) | 4-16 |
| 4-15 | Level Three and Level Four Statistical Results (Mission Area) | 4-17 |
| 4-16 | Level Three and Level Four Statistical Results (Software Class) | 4-17 |
| 4-17 | Level Three and Level Four Statistical Results (Software Status) | 4-18 |
| 4-18 | Level Three and Level Four Statistical Results (Software Mode) | 4-19 |
| 4-19 | Level Three and Level Four Statistical Results (Language) | 4-20 |
| 4-20 | Level Three and Level Four Statistical Results (Size) | 4-20 |
| 4-21 | Level Three and Level Four Statistical Results (Summary) | 4-21 |
| 5-1 | Arriving at the NCCA Normalized Database..... | 5-2 |
| 5-2 | Normalized Database by Source Database..... | 5-3 |
| 5-3 | Key Aspects of Remaining Source Databases..... | 5-3 |
| 5-4 | Summary of Data Partitions..... | 5-4 |
| 5-5 | Summary of Regressions..... | 5-8 |
| 5-6 | Subset Distribution of Data Points Across Source Databases..... | 5-12 |
| 5-7 | Summary of Validation Database..... | 5-20 |
| 5-8 | Summary of Program versus CSCI Differences..... | 5-21 |
| 5-9 | Program-Level Equations Remaining..... | 5-24 |
| 5-10 | CSCI-Level Equations Remaining..... | 5-24 |
| 6-1 | Productivity Factor (MIS Programs) | 6-3 |
| 6-2 | Productivity Factor (30% Assembly Programs) | 6-4 |
| 6-3 | Productivity Factor (100% Assembly Programs) | 6-5 |
| 6-4 | Productivity Factor (Physical Programs) | 6-6 |
| 6-5 | Productivity Factor (Unknown Code Condition Programs) | 6-7 |

| | | |
|-------|---|-------|
| 6-6 | Productivity Factor (Logical Programs, Phasing Unknown) | 6-8 |
| 6-7 | Summary of Top-Level Productivity Factors..... | 6-9 |
| 7-1 | Regression versus Factor Performance..... | 7-4 |
| 8-1 | NCCA Normalized Schedule Database..... | 8-4 |
| 8-2 | NCCA Normalized Schedule Database Partitions..... | 8-5 |
| 8-3 | Statistical Results of Partitions..... | 8-5 |
| 8-4 | Comparison of Equation [8-2] to Other Traditional Schedule Estimation Models..... | 8-7 |
| 9-1 | NCCA Raw Software Labor Rate Database..... | 9-5 |
| 9-2 | NCCA Normalized Software Labor Rate Database FY97\$K (Cost through G&A)..... | 9-6 |
| 9-3 | Average Labor Rate Analysis (Cost through G&A) | 9-9 |
| 9-4 | Nonparametric Analysis..... | 9-9 |
| 10-1 | Mean Percent SLOC Growth Analysis..... | 10-4 |
| 10-2 | Median Percent SLOC Growth Analysis..... | 10-5 |
| 10-3 | Summary of Approach Two - Size Growth Estimating Relationships..... | 10-6 |
| 10-4 | Residuals of Actual Total SLOC versus Estimated Total SLOC..... | 10-7 |
| 10-5 | Programs with Reuse..... | 10-8 |
| 10-6 | Approach Three Size Growth Estimating Relationships..... | 10-8 |
| 10-7 | Software Growth Estimating Relationships..... | 10-9 |
| 10-8 | Summary of the Statistics for SLOC Growth Methodology..... | 10-9 |
| 10-9 | Code Condition Mean Percentage Statistics..... | 10-11 |
| 10-10 | Code Condition Median Percentage Statistics..... | 10-12 |
| 10-11 | Code Condition Mean Percentage Point Statistics..... | 10-13 |
| 10-12 | Code Condition Median Percentage Point Statistics..... | 10-13 |
| 10-13 | Approach Two - Code Condition Estimating Relationships Results..... | 10-14 |
| 10-14 | Summary of Statistics for Code Condition Methodology..... | 10-14 |

List of Figures

| | | |
|------|--|-------|
| 1 | Software Development Estimating Process..... | x |
| 1-1 | Software Development Estimating Process..... | 1-1 |
| 3-1 | Source Code Example..... | 3-3 |
| 3-2 | Example of Software Development Across the Acquisition Phases..... | 3-5 |
| 3-3 | Software Development Phases and Reviews..... | 3-14 |
| 3-4 | New Code versus Common Code..... | 3-20 |
| 4-1 | Mission Data Sets..... | 4-3 |
| 4-2 | Counting Convention Data Set..... | 4-6 |
| 4-3 | Language Data Set..... | 4-8 |
| 4-4 | Phases of Software Development..... | 4-9 |
| 4-5 | Phasing Data Set..... | 4-10 |
| 4-6 | Level Three and Level Four Data Sets..... | 4-14 |
| 5-1 | Trace of Standard Error and Predict (20) | 5-6 |
| 5-2 | Predict (20) Tradeoff..... | 5-7 |
| 5-3 | One Hundred Percent New CSCIs..... | 5-13 |
| 5-4 | Effort Discount as a Function of Reuse..... | 5-17 |
| 5-5 | Average CSCI Size versus CSCI-Program Differences..... | 5-22 |
| 5-6 | Percent New versus CSCI-Program Differences..... | 5-22 |
| 5-7 | Program Size versus CSCI-Program Differences..... | 5-23 |
| 5-8 | CSCI Count versus CSCI-Program Differences..... | 5-23 |
| 7-1 | Effort Estimating Process..... | 7-1 |
| 8-1 | NCCA Schedule Databases..... | 8-3 |
| 8-2 | Recommended Software Schedule Estimating Process..... | 8-8 |
| 9-1 | Contractor Cost Data Report, 1921..... | 9-2 |
| 9-2 | Functional Cost-Hour Report, 1921-1..... | 9-3 |
| 9-3 | Cost Performance Report, Format 1..... | 9-4 |
| 9-4 | Cost Performance Report Manpower Loading Report, Format 4..... | 9-4 |
| 9-5 | Recommended Labor Rate Estimation Process..... | 9-12 |
| 10-1 | Recommended Risk Analysis Process..... | 10-16 |

EXECUTIVE SUMMARY

INTRODUCTION

The Naval Center for Cost Analysis (NCCA) organized an in-house software team of six analysts to assess NCCA's software cost estimating process. The team discovered that NCCA did not have a well defined and consistent process for estimating software cost. NCCA was using Software Architecture Sizing & Estimating Tool (SASET) and Revised Intermediate Constructive Cost Model (REVIC) to perform estimates, but in-house guidance on the correct procedure to develop a software estimate did not exist. After several discussions regarding the lack of consistency in how NCCA used these models, the team conducted a survey to determine if other governmental agencies and contractors were experiencing the same problems. The survey was disseminated to a total of 25 governmental agencies and contractors. The survey responses indicated that other organizations were experiencing similar problems, so NCCA decided to conduct an extensive research effort to improve in-house (and hopefully other organizations') software cost estimating capabilities.

OBJECTIVE and SCOPE

The NCCA software team's mission was:

"To provide the individual analyst with the procedures, tools, and training to develop a defensible and reproducible software life cycle cost estimate."

NCCA developed goals to ensure that the mission was adhered to and accomplished. There were two phases to this software research effort. Phase One was the baseline software development cost research effort. It is this effort which is documented in this handbook. The Phase One goals were to provide: 1) a centralized and well documented database comprised of existing software databases, 2) formal procedures and guidelines for developing a software estimate, 3) top-level software estimating tools, and 4) training. Although the mission of the software team was to address the software life cycle, the maintenance phase was not addressed in Phase One. The goals for Phase Two are to collect and summarize all the documentation on existing commercial software cost models and to focus on data collection efforts, especially in the Automated or Management Information System (AIS/MIS) domain.

METHODOLOGY and RESULTS

NCCA developed a five-step software development estimating process as detailed in Figure 1. The handbook is organized according to this process.

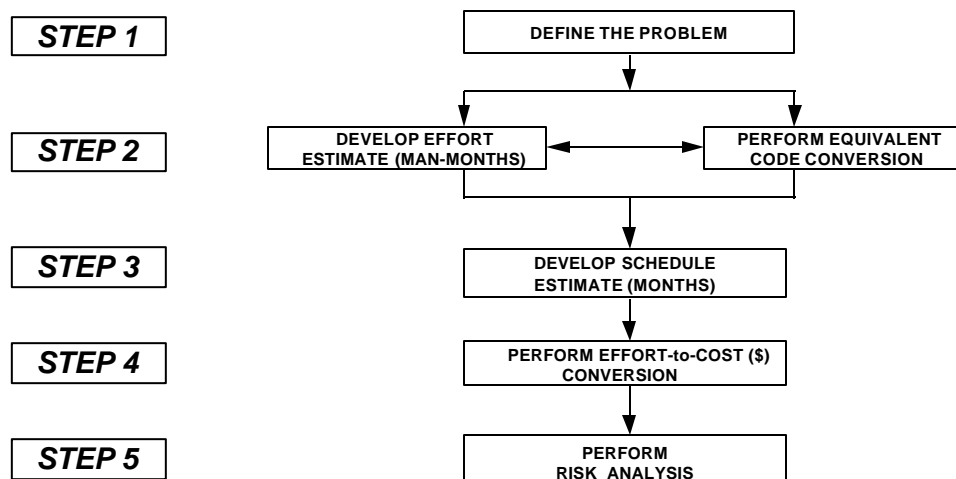


Figure 1: Software Development Estimating Process

The key findings and results (i.e., tools) associated with each step are summarized below. NCCA contends that the analytical approach used to develop the standard tools (vice the tools themselves) is the most important aspect of this effort. Therefore, NCCA strongly recommends that the standard tools not be utilized before the analyst has completely read and understood: 1) the approach utilized to develop the tools and 2) the tools' strengths and weaknesses. This document includes a significant amount of detail regarding the team's data normalization and analysis process that should facilitate this understanding.

Step 1: Defining the Problem

NCCA developed a standard form, "NCCA Software Program Definition Form", and an associated data field dictionary, which can be utilized to obtain information on the program being estimated. Since the Phase One analysis is geared toward the novice software cost estimator, most of the form targets objective metrics predicated on the results of the analytical efforts documented herein. Additionally, NCCA developed the form, "NCCA Historical Software Data Request Form", and an associated data field dictionary, to aid in the collection of historical data. Finally, NCCA developed the "NCCA Historical Software Data Request Form's Mapping Procedures" form, which documents the mapping procedures that should be utilized when entering newly obtained historical data into the NCCA Raw Software Effort Database (discussed in Section 3 - **Software Database**).

Step 2: Effort Estimation/ESLOC Conversion

NCCA Raw Software Effort Database:

Prior to beginning the analytical efforts, NCCA developed a software database. For Phase One, the NCCA Raw Software Effort Database drew upon data currently available to NCCA. The NCCA Raw Software Effort Database consists of 457 unique records or data points; 151

program-level and 306 Computer Software Configuration Item (CSCI)-level¹ compiled from eight different databases:

- 1) MITRE Non-Ada Database
- 2) MITRE Ada Database
- 3) Space and Missile Center (SMC) Database
- 4) NASA Software Engineering Laboratory (SEL) Database
- 5) Navy (NCCA) Internal Database
- 6) SASET Validation Database
- 7) REVIC Recalibration Database
- 8) IIT Research Institute (IITRI) Database

The database contains 73 attribute fields, however, all fields are not completed for each record. These fields describe various attributes of the program including size, effort, schedule, language, and development process. At a minimum, size and effort are provided for each of the 457 data points. The programs were developed from the early 1970s through the 1990s.

NCCA Normalized Software Effort Database:

The NCCA Raw Software Effort Database does not support meaningful analyses. Many of the records in the database have different units of measure for items such as size, effort and schedule. NCCA contends that there are two possible approaches to arrive at a normalized database. The first approach is to filter out any data point that does not meet specified criteria (e.g., exclude all data points which do not include the software requirements phase). A second approach is to keep all of the data points, but adjust them as necessary in order to obtain consistent units of measure (e.g., adjust all data points which don't include the requirements phase). The latter approach can be extremely subjective; adjustments can vary tremendously depending upon the methodology chosen (engineering judgment versus historical data). Therefore, in order to minimize the amount of uncertainty introduced into the data, NCCA chose to normalize the database by filtering out those data points which did not satisfy the specified criteria (i.e., no factors were used to normalize data points).

To properly normalize the database, NCCA attempted to isolate those objective software metrics which significantly drive productivity by stratifying the data and conducting statistical non-parametric tests. The following software metrics were proven to be statistically significant when estimating software productivity:

- 1) Mission (Domain) – MIS versus Weapon System
- 2) Counting Convention - Physical versus Logical
- 3) Language - Assembly versus High Order Language (HOL (e.g., FORTRAN, Jovial))
- 4) Phasing - Software Specification Review (SSR) through Formal Qualification Test (FQT) versus System Design Review (SDR) through FQT
- 5) Code Condition – percent (%) new, % reused (e.g., modified, verbatim, translated, rehosted, etc.)
- 6) Development Mode - Embedded versus Non-Embedded

¹ There were a total of 329 CSCI-level data points; 306 CSCI-level data points plus 23 program-level data points. These 23 program-level data points contained only one CSCI.

Based on these findings, the NCCA Normalized Software Effort Database included only those data points that met the following criteria:

- 1) Mission or Domain was weapon system
- 2) Counting Convention was logical
- 3) Development language was HOL
- 4) Development phases span SDR through FQT
- 5) Code condition was known
- 6) Development mode was known
- 7) Effort in hours or hours per man-month were known (and normalized to 152 hours per man-month).

As depicted in Table 1 below, the normalization procedure eliminated most of the data points included in the raw database.

| | Initial Number of Data Points | |
|---|---------------------------------|-----------------------|
| | Program | CSCI |
| Start: Top-Level | 151 | 329 ² |
| ⤵ | | |
| Normalizing Factors | Number of Data Points Remaining | |
| | Program | CSCI |
| Mission = Weapon System | 105 | 236 |
| ⤵ | | |
| Code Count = Logical | 56 | 185 |
| ⤵ | | |
| HOL ≥ 70% | 47 | 146 |
| ⤵ | | |
| Scope of Effort = SDR through FQT | 32 | 100 |
| ⤵ | | |
| Code Condition Known | 31 | 97 |
| ⤵ | | |
| Development Mode Known | 31 | 97 |
| ⤵ | | |
| Hours/man-month = known | 31 | 97 |
| Final NCCA Normalized Software Effort Database | 31 | 97³ |

Table 1: Evolution of the NCCA Normalized Software Effort Database

The final program-level NCCA Normalized Software Effort Database consists of 31 data points. The start dates were not provided for all data points. However, the start dates provided were from 1972 through 1984. These software developments were written in FORTRAN, Ada, and JOVIAL. The SLOC range is from 9 to 1,113 KSLOC. The total effort ranged from 9 to 10,976 man-months. A majority of the program-level data points are semi-detached, while some embedded and organic modes are represented. This database includes various missions, such as: radar, command, control and communications (C³), and simulation, which were installed on both ground and ship platforms.

The final CSCI-level NCCA Normalized Software Effort Database consists of 97 data points. Similar to the program-level database, the start dates were not provided for all data points. However, the CSCI start dates provided were from 1972 through 1991. These software developments were written in FORTRAN, Ada, CMS-2, JOVIAL, ATLAS and C. The SLOC range is from 0.411 to 492 KSLOC. The total effort ranged from 2.1 to 5,007 man-months. A

²Three hundred and six CSCI-level data points plus 23 program-level data points that contained only one CSCI.

³Ninety-three CSCI-level data points plus four program-level data points that contained only one CSCI.

majority of the CSCI-level data points are embedded, while some semi-detached and organic modes are represented. This database includes various missions, such as: radar, Anti-Submarine Warfare (ASW), C³, simulation and missile, which were installed on ground, air and ship platforms.

Table 2 shows which source databases remained after normalization and why the other data sources were deleted.

| Source Database | Code | Number of Data Points | | Reason for Database Exclusion |
|---------------------|------|-----------------------|------|--|
| | | Program | CSCI | |
| MITRE Non-Ada | 1 | 13 | 38 | |
| MITRE Ada | 2 | 0 | 0 | Effort did not reflect SDR through FQT |
| SMC | 3 | 4 | 6 | |
| NASA SEL | 4 | 14 | 4 | |
| Navy Internal | 5 | 0 | 45 | Program-level data points utilized physical SLOC counting convention |
| Silver SASET | 6 | 0 | 0 | Code count, hours/man-month, and scope of effort unknown |
| REVIC Recalibration | 7 | 0 | 0 | Hours/man-month for data points from non-SMC sources could not be verified |
| IITRI | 8 | 0 | 4 | Did not contain program-level data |
| TOTAL | | 31 | 97 | |

Table 2: NCCA Normalized Software Effort Database and Source Databases

Analysis:

NCCA's recommended approach to estimating software development effort is to use contractor-specific data. However, since the majority of the data in the NCCA Normalized Software Effort Database was previously sanitized (non-program or non-contractor-specific) by the source database developer, NCCA developed standard normalized estimating relationships. The normalized estimating relationships estimate effort in man-months (MM) as a function of size. In this handbook, the size metric used is equivalent new source lines of code (ESLOC). ESLOC are the weighted sum of new code and reused code. It is generally accepted that adapted SLOC do not require the full software development effort (design, code, test and documentation). The ESLOC conversion is typically based on engineering judgment, however, NCCA developed a unique quantitative approach to ESLOC calculation. The normalized estimating relationships (which also perform ESLOC calculations) should be utilized if and only if the program being estimated meets the normalization criteria set forth previously and contractor-specific data does not exist.

In the event the program being estimated does not meet the normalization criteria, NCCA developed non-normalized top-level productivity factors from the NCCA Raw Software Effort Database. While the variances associated with the normalized standard regressions are large, those associated with the non-normalized productivity top-level standard factors are even larger. NCCA believes that the magnitude of these variances can be largely attributed to the fact that the standard regressions and factors reflect industry averages. NCCA believes developing contractor-specific tools can reduce these variances.

NCCA then analyzed and compared the resulting statistics and associated performance parameters of the regressions developed. Based on this analysis, the recommended software effort estimating methodology developed, and documented within this handbook, should be applied as follows.

If the program being estimated satisfies all of the following criteria:

- 1) Mission or Domain is weapon system
- 2) Counting Convention is logical
- 3) Development language is HOL
- 4) Development phases span SDR through FQT
- 5) Code condition is known
- 6) Development mode is known
- 7) Effort in hours or hours per man-month is known

then, the NCCA **Normalized Regressions** should be utilized as follows:

- If the program being estimated is 100 percent new, apply the following equation **at the program-level**:

$$\text{Effort (MM)} = 0.0012 * (\text{New SLOC})^{[1.1979 + (0.0326 * D_1)]}$$

$R^2 = 0.96$; Std Error = 0.42; Predict (20)⁴ = 58%; n = 31; Range = 9 - 1,113 EKSLOC
where the dummy variable, D_1 , equals one if the program is embedded and zero otherwise.

- If the program being estimated is equal to or greater than 82 percent reused code, apply the following equation **at the program-level**:

$$\text{Effort (MM)} = 0.0012 * [\text{New SLOC} + (1 * \text{Reused SLOC})]^{[1.0085 + (0.0326 * D_1)]}$$

$R^2 = 0.96$; Std Error = 0.42; Predict (20) = 58%; n = 31; Range = 9 - 1,113 EKSLOC
where the dummy variable, D_1 , equals one if the program is embedded and zero otherwise.

- If the program being estimated has less than 82 percent reused code which is **evenly distributed between modified and verbatim code**, apply the following equation **at the program-level**:

$$\text{Effort (MM)} = 0.0012 * [\text{New SLOC} + (1 * \text{Reused SLOC})]^{[1.1067 + (0.0326 * D_1)]}$$

$R^2 = 0.96$; Std Error = 0.42; Predict (20) = 58%; n = 31; Range = 9 - 1,113 EKSLOC
where the dummy variable, D_1 , equals one if the program is embedded and zero otherwise.

- If the program being estimated has less than 82 percent reused code which is **predominantly verbatim**, apply the following equation **at the CSCI-level**:

$$\text{Effort (MM)} = 0.023 * [\text{New SLOC} + (0.03 * \text{Reused SLOC})]^{[0.8609 + (0.0529 * D_1)]}$$

$R^2 = 0.77$; Std Error = 0.67; Predict (20) = 26%; n = 97; Range = 0.4 - 253.4 EKSLOC
where the dummy variable, D_1 , equals one if the program is embedded and zero otherwise.

If the program being estimated does not satisfy NCCA's normalization criteria, then the NCCA Standard **Non-Normalized Productivity Factors** should be utilized as follows:

⁴ Predict (20) is the percentage of time the total residuals are within 20 percent of the actual value. See Appendix C for more details on Predict (20) calculations.

- If the program is MIS and
 - 1) Code condition is unknown = **0.6913 Hrs/Total SLOC**
Coefficient of Variation (CV) = 86%; CV_{est} = 132%; n = 17
 - 2) Code condition is known = **0.8240 Hrs/ESLOC**
Efactor = 0; CV = 72%; CV_{est} = 103%; n = 17
- If the program is written entirely in Assembly and
 - 1) Code condition is unknown = **2.6504 Hrs/Total SLOC**
CV = 120%; CV_{est} = 177%; n = 68
 - 2) Code condition is known = **3.0093 Hrs/ESLOC**
Efactor = 0.6; CV = 115%; CV_{est} = 168%; n = 61
- If the program is written significantly (>30%) in Assembly and
 - 1) Code condition is unknown = **3.7383 Hrs/Total SLOC**
CV = 100%; CV_{est} = 132%; n = 40
 - 2) Code condition is known = **3.9904 Hrs/ESLOC**
Efactor = 0.69; CV = 98%; CV_{est} = 125%; n = 38
- If the counting convention is physical and
 - 1) Code condition is unknown = **0.6357 Hrs/Total SLOC**
CV = 124%; CV_{est} = 93%; n = 18
 - 2) Code condition is known = **0.7350 Hrs/ESLOC**
Efactor = 0; CV = 104%; CV_{est} = 123%; n = 18
- If the counting convention is unknown and
 - 1) Code condition is unknown = **1.3238 Hrs/Total SLOC**
CV = 128%; CV_{est} = 196%; n = 273
 - 2) Code condition is known = **1.6763 Hrs/ESLOC**
Efactor = 0.12; CV = 107%; CV_{est} = 215%; n = 262
- If the counting convention is logical, but phasing is unknown and
 - 1) Code condition is unknown = **1.3360 Hrs/Total SLOC**
CV = 113%; CV_{est} = 182%; n = 186
 - 2) Code condition is known = **1.8597 Hrs/ESLOC**
Efactor = 0.04; CV = 83%; CV_{est} = 161%; n = 185

NCCA used a validation data set to demonstrate and compare the performance of the regressions and productivity factors; the **regressions greatly outperform the productivity factors, therefore, NCCA strongly encourages the analyst to define the program being estimated according to the normalization criteria identified previously.** For example, if you have a choice of counting conventions, choose logical so that the normalized regressions, vice the non-normalized productivity factors, can be used.

Step 3: Schedule Estimation

NCCA Raw Schedule Database:

NCCA performed a query of the NCCA Raw Software Effort Database to obtain data points that included schedule as well as effort. The query resulted in a total of 151 program-level data points. These 151 data points were screened to identify those having schedule dates from SDR through FQT. Thirty-seven of the 151 points met this criterion and were retained. These data points constitute the NCCA Raw Schedule Database.

NCCA Normalized Schedule Database:

The 37 data points in the NCCA Raw Schedule Database were then screened further to obtain the NCCA Normalized Schedule Database. The additional criteria for the NCCA Normalized Schedule Database were:

- 1) Effort reflects SDR through FQT
- 2) Effort in hours or hours per man-month were known (and converted to 152 hours per man-month).

The resulting 16 data points are a mixture of HOL and Assembly language programs. However, none of the HOL programs were written in Ada. The mission types are characterized as C³, radar, and simulation programs that are installed on air, ship, and ground platforms. The total SLOC ranges from 20 to 1,113 KSLOC, and total development effort ranges from 157 to 10,976 man-months. The associated schedules range from 12 to 74 months. The mean schedule is 33.3 months.

Analysis:

Similar to the effort analysis, NCCA created normalized top-level factors and schedule estimating relationships where schedule is a function of effort in man-months or size in ESLOC. The final recommended tool, applicable to weapon system programs with development phases spanning from SDR through FQT, is:

$$\text{Schedule (Months)} = 5.12 (\text{MM})^{0.2266} * e^{(0.3574 * D_1)}$$

R² = 0.64; Std Error = 0.31; Predict (20) = 44%; n = 16; Range = 157 – 10,976 MM
where the dummy variable, D₁, equals one if the program is 100 percent new and zero otherwise.

Step 4: Effort-to-Cost Conversion

NCCA Raw Labor Rate Database:

NCCA collected software cost and manning data from 34 weapon system programs. The data was collected from both cost performance reports (CPRs) and contractor cost data reports (CCDRs). The raw software cost data is in then-year dollars and includes general and administrative cost (G&A). The manning data is expressed in man-hours or man-months.

NCCA Normalized Labor Rate Database:

After scrutiny of the data points, some programs were excluded from the database for one of the following reasons:

- 1) Man-hours were not reported.
- 2) Development phases didn't reflect SDR-FQT.
- 3) Major software development problems occurred (i.e., flight test failures) which implied that these points were outliers.
- 4) Programs were less than 90 percent complete.

The resulting NCCA Normalized Labor Rate Database has 15 programs. The man-hours expended to develop the software range from 2K to 793K and the software costs through G&A range from \$317K to \$95M in FY97\$. The NCCA Normalized Labor Rate Database consists of aircraft, ships, missiles and electronics programs, representing both cost-plus and fixed-price contracts and East and West Coast contractors. The first year of development ranged from 1982 to 1992. NCCA did not collect data for MIS programs.

Analysis:

NCCA developed top-level factors for the different populations, in order to conduct non-parametric tests. Based on the non-parametric tests, NCCA determined that platform type (i.e., aircraft versus non-aircraft) was a statistically significant independent variable. Hence, NCCA developed two types of regressions: with and without dummy variables. NCCA then analyzed and compared the resulting statistics and associated performance parameters of the factors and regressions developed. Based on the analysis, NCCA recommends the following regression be used to convert effort to cost if contractor-specific data is not available:

Cost through G&A

$$\text{FY97\$K} = 136.93 * (\text{Labor KHrs})^{0.98} * e^{(-0.40 * D_1)}$$

$R^2 = 0.99$; Std Error = 0.13; Predict (20) = 87%; n = 15; Range 2 - 793 Labor KHrs
where the dummy variable, D_1 , equals one if the program is non-aircraft and zero otherwise.

NCCA also developed a regression for a fully burdened estimate. The price data points utilized in the regression were estimated by applying average Cost of Money (COM) and fee rates to the cost through G&A data points in the NCCA Normalized Labor Rate Database. For those programs where fee was zero or not known, an average rate based on the available data points was used.

Price

$$\text{FY97\$K} = 154.21 * (\text{Labor KHrs})^{0.98} * e^{(-0.39 * D_1)}$$

$R^2 = 0.99$; Std Error = 0.13; Predict (20) = 87%; n = 15; Range 2 - 793 Labor KHrs
where the dummy variable, D_1 , equals one if the program is non-aircraft and zero otherwise.

Step 5: Risk Analysis

NCCA Raw Risk Analysis Database:

NCCA performed a query of the SMC Software Database to obtain data points that included both actual and estimated SLOC. The query produced 12 program-level and 28 CSCI-level data points. Next, a search of NCCA's files provided 10 additional program-level data points. Finally, one program-level and four CSCI-level data points were collected from an Institute for Defense Analyses (IDA) study for a total of 23 program-level and 32 CSCI-level data points.

NCCA Normalized SLOC Growth Database:

The NCCA Raw SLOC Growth Database was screened, as follows, to arrive at the normalized database:

- 1) Based on the Mann Whitney U test and the Kolmogorov-Smirnov test, the CSCI-level data (32 data points) was deleted. The tests showed that the means and variances of CSCI and program-level data points were not equal.
- 2) Four program-level data points were eliminated because the initial SLOC estimates could not be verified.
- 3) Three MIS programs were excluded in order to remain consistent with the effort analysis results.

This screening resulted in 16 program-level data points. Although NCCA is confident the software for these programs was developed between Milestones II and III, specific review dates are unknown. The majority of the program names are also unknown. The range of estimated SLOC values are 14 to 1,246 KSLOC; nine programs are less than 100 KSLOC. Five programs are entirely new. All were weapon system programs whose code condition, both new and reused, was known.

Analysis:

NCCA addressed two areas of risk in the software development estimating process. First, NCCA addressed software development growth rates that reflect the difference between the estimated and the actual lines of code. The purpose of this effort was to provide a risk assessment of the initial software lines of code estimate.

Second, NCCA addressed software code condition risk. Often, the initial sizing estimate reflects an overestimate of the amount of reused code. NCCA developed a scheme for redistributing code estimates from reused to new.

NCCA recommends that these two risk areas be applied as follows:

- 1) To estimate SLOC growth risk, add 22 percent to the initial total SLOC sizing estimate.

- 2) To estimate code condition risk, add eight percentage points to the initial percent new estimate (if less than 93 percent) and subtract eight percentage points from the initial percent reused estimate.

CONCLUSIONS

NCCA accomplished the Phase One mission by providing NCCA analysts with a normalized database and associated top-level software development estimating tools, which are credible, reproducible, defensible and well documented. This handbook will equip a cost analyst with the proper techniques for understanding, developing and utilizing objective software development estimating tools. Unfortunately, though not surprisingly, the resulting effort, schedule and risk tools exhibit significant variability, as evidenced by the CVs which are typically higher than 30 percent. This variability is likely due to a variety of factors (e.g., different complexity levels or various development processes), but NCCA believes the variability can be largely attributed to the fact that the underlying databases reflect the capabilities of a mix of contractors. We highly recommend that, whenever possible, the analyst strive to develop software estimates that are based on contractor-specific data. For those software estimating tasks where historical, contractor-specific data is not available, the standard (i.e., industry average) tools presented in this handbook are appropriate. For those estimating tasks where historical, contractor-specific data is available, the handbook should be used as a “how-to” guide to develop contractor-specific tools.

While the tools presented in this handbook have their documented limitations, NCCA views the handbook in a broader sense as a procedural and analytical framework for continual software database and methodology improvement.

ACKNOWLEDGMENTS

The ideas, approaches, and conclusions presented in this handbook reflect much more than the experience and research of the authors. The numerous references throughout the handbook show the extent to which we have used published research and discussions with knowledgeable experts who have been involved in software development. It is only through this open exchange of information that this handbook was made possible, and only through continued conversation that the handbook can continue to evolve. Therefore, NCCA welcomes and encourages any and all comments.

The authors are especially grateful to Mr. Lowell Blagmon for his contributions to Section 8 - **Schedule Analysis** and Mr. Rick Collins for his invaluable guidance and assistance. Thanks are also extended to the following NCCA employees for reading and editing the handbook: Mr. Jack Smuck, Mr. Brian Flynn, Mr. Stephen Gross, Ms. Nancy St. Louis, Ms. Karen Richey, Mr. John Georges, LCDR Katherine Kinnavy, and Mr. Jeff Cherwonik. The authors also appreciate the review conducted by Mr. Gene Waller of Technomics, Inc..

INTRODUCTION

1.1 HANDBOOK INTRODUCTION

Although a multitude of software development estimating tools currently exist, software development is still one of the most difficult areas to estimate. The Naval Center for Cost Analysis (NCCA) contends that only by targeting specific contractors and their associated development processes (including personnel, tools and methodologies) can cost estimators expect to decrease the large variances associated with software development estimates. A detailed, contractor-specific database and associated contractor-specific software development estimating tools do not currently exist within the Navy. This handbook is not intended to fill this void, but to provide the analyst with a set of standard instructions and tools to utilize when additional contractor-specific, analogous data is unavailable. These tools will allow the analyst to develop a comprehensive, defensible and reproducible software development estimate, with the associated statistical variances defined. Although this document is intended for novice software cost estimators, it is not tutorial in nature. However, through close attention to the assumptions and analytical approaches underlying the standard software development estimating tool set, the analyst should acquire many of the skills necessary to develop or analyze other software development estimating approaches. Additionally, the specific strengths and weaknesses of the recommended approaches, including examples and possible follow-on research efforts, are discussed to facilitate usage of, and modifications and improvements to, the standard tools.

NCCA recommends a five-step software development cost estimating process, as illustrated in Figure 1-1:

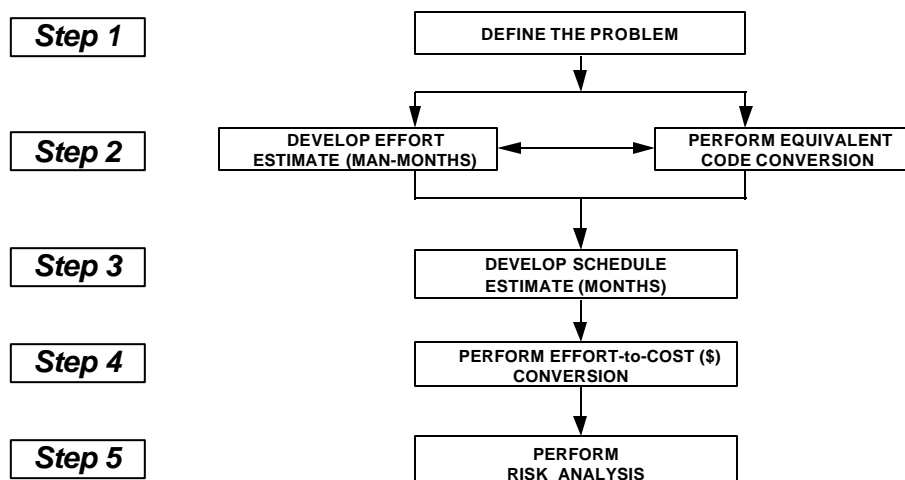


Figure 1-1: Software Development Estimating Process

For each major step of the process, this document discusses: 1) the underlying database, 2) the normalization efforts performed, 3) the standard regressions or factors developed, (with their associated strengths and weaknesses), and 4) the final recommended methodology, including detailed instructions which address when and how to apply the tools. Proper use of these tools requires the analyst to understand the weaknesses in the underlying database in order to quantify, rectify, or at a minimum, qualify the impacts of these weaknesses for the program being estimated. As mentioned earlier, these standard tools are to be used only when additional, contractor-specific, analogous data is unavailable. This document discusses, in detail, why contractor-specific data is essential to develop the most accurate software development estimate; and in the event that contractor-specific data is obtained, this document will serve as a detailed guide to the processes required to normalize and analyze this data and the resulting tools.

1.2 HANDBOOK OVERVIEW

Including the introduction (**Section 1**), this document consists of 11 sections.

Section 2 of this document addresses the first step of the software development estimating process: **Defining the Problem**. It provides an overview of the standard data definition forms available to gather the essential technical and programmatic data required to develop a software development estimate.

Sections 3 through 7 discuss the second step of the software development estimating process: **Effort Analysis**. Specifically, **Section 3 - Software Database**, addresses the data collection methods utilized to develop the NCCA Raw Software Effort Database, including the source databases, procedures and schema. Additionally, because technology is evolving so quickly in the software development area, this section references several separate issue papers that should be taken into consideration when developing a software estimate. These issues, such as the impact of tailoring military standards (MIL-STDs) or the use of Commercial-Off-The-Shelf (COTS) software, are not represented in the NCCA Raw Software Effort Database and may ultimately affect the effort estimate. Where data allowed, quantitative adjustments are provided to account for these types of advancements. Otherwise, positions based on the qualitative assessment of various experts' opinions are provided and recommended to be used until supporting quantitative data becomes available. **Section 4 - Effort Analysis: Significant Drivers** documents the analytical procedures followed to determine software development productivity drivers. NCCA's general goal was to identify the significant, objective, software development metrics which most affect productivity. Due to the intended audience, NCCA wanted to eliminate as much subjectivity as possible when developing a software estimate. A discussion concerning equivalent code conversion techniques is also provided. **Section 5 - Effort Analysis: Normalized Regressions** reviews the procedures followed to develop the final normalized database and the analytical approach utilized to develop the standard normalized set of software development estimating tools. An evaluation of the resulting tools on a validation database is also provided. **Section 6 - Effort Analysis: Non-Normalized Productivity Factors** provides the analyst with a standard set of tools in the event the program being estimated does not meet NCCA's normalization criteria.

Section 7 - Effort Analysis: Overall Process presents the overall software development effort estimating process and an example of the resulting statistics when applied to a sample population of programs.

Section 8 - Schedule Analysis addresses the third step of the software development estimating process: **Schedule Estimation**. This section documents the schedule databases and associated ground rules and assumptions for both the raw and normalized databases, and details the analytical approach followed to derive top-level factors and schedule estimating relationships, along with the tools' associated strengths and weaknesses.

Section 9 - Labor Rate Analysis addresses the fourth step of the software development estimating process: **Effort-to-Cost Conversion (Man-Year Rate) Estimation**. This section documents the labor rate databases and associated ground rules and assumptions for both the raw and normalized databases, as well as details the analytical approach followed to derive top-level factors and cost (man-year rate) estimating relationships, along with their associated strengths and weaknesses. Guidance is also provided on adjustment techniques to best reflect the current business base and acquisition strategy of the corporation.

Section 10 - Risk Analysis discusses the fifth step of the software development estimating process. It provides NCCA's recommended approach for evaluating the risk associated with the software development effort, including source lines of code (SLOC) growth and estimated percent reused changes.

Section 11 - Conclusions provides final remarks for the Phase One Handbook.

Finally, a list of references, acronyms and supporting Appendices are provided.

DEFINING THE PROBLEM

There are two basic types of information required to develop a quality software development estimate: 1) technical and programmatic information for the program being estimated and 2) technical and programmatic information for the analogous historical programs used to develop the estimate.

Based on NCCA's analysis, a software development estimate requires, at a minimum, the following information for the program being estimated:

- Some measure of the work to be performed with associated units (i.e., SLOC counts, words, function points, etc.)
- If SLOC is utilized as the unit of measure, the associated counting convention (i.e., physical, physical with comments, logical, etc.)
- The programming language utilized (at a minimum Assembly versus Higher Order Languages (HOL (e.g., FORTRAN, Jovial, CMS-2, etc.)) versus Fourth Generation Languages (4GL))
- The condition of the code (i.e., percent new, percent reused (modified, verbatim, translated, rehosted, etc.)), with associated definitions
- The phases of the software development life cycle to be estimated (e.g., System Design Review (SDR) through Formal Qualification Test (FQT))
- The development mode (at a minimum, embedded versus non-embedded)
- If known, the name of the contractor responsible for developing the program. As discussed previously, NCCA contends that contractor-specific data holds the greatest possibility for increasing the accuracy and decreasing the variance associated with the software estimating tools developed.

Sections 3 through 6 provide definitions of the variables cited above, and discuss in detail how NCCA arrived at this list and the importance of obtaining each of the required inputs. NCCA has developed a standard **NCCA Software Program Definition Form** with an associated **Data Dictionary** to facilitate the collection of the information cited above. See Appendix A for these documents. The **NCCA Software Program Definition Form** requests information in addition to that cited in the list above to support future improvements to this handbook. Since all of the information requested affects the projected productivity of the development effort, it is crucial that the information gathered be as specific as possible.

In addition to the aforementioned information on the program being estimated, the analyst must compile the same information for the analogous contractor-specific historical programs that will be used to develop the cost estimating methodology. Additionally, the actual effort, schedule, and cost (price) to develop the software, by software development phase if possible, should be obtained. NCCA has also developed a standard **NCCA Historical Software Data Request Form** with an associated **Data Dictionary** to standardize the collection of historical data. See Appendix A. It is with this information that the most accurate productivity, schedule, and labor rate metrics can be developed. If the Request for Proposal (RFP) was developed correctly, the Software Development Plan (SDP) Contract Data Requirements List (CDRL) is an excellent source of historical data. The SDP typically requires a list of previously delivered programs developed by the contractor, with the associated technical and programmatic data. If, however, the SDP is not available, this type of information can and should still be obtained from the contractor in whatever form is available. When collecting historical data, the analyst must ensure that the information is for completed programs. Often, projections of on-going efforts are mixed in with actual completed programs. Since software development is continuously evolving, the analyst should **always** try to obtain the most recent data available. Thus, the regressions and factors presented within this document can be continuously updated to capture the latest technological trends.

Finally, NCCA has documented the procedures the in-house analyst should follow when entering new data into the NCCA Raw Software Effort Database. Appendix A also includes the **NCCA Historical Software Data Request Form's Mapping Procedures**.

SOFTWARE DATABASE

3.1 INTRODUCTION

This section of the handbook documents the source databases, procedures and schema used to create the NCCA Raw Software Effort Database. The NCCA Raw Software Effort Database (hereafter called the NCCA Raw Database) is a conglomeration of several historical software databases from both internal NCCA and external sources. The NCCA Raw Database drew upon data currently available to NCCA and was designed for a top-level software cost analysis. NCCA defines a Phase One analysis as one that relies on basic objective inputs. It uses either top-level productivity factors or estimating relationships (equations derived using least squares regression) to estimate the effort and schedule of a software program.

The NCCA Raw Database contains a total of 457 records from many different Department of Defense and National Aeronautics and Space Administration (NASA) software development programs at both the program and CSCI-level. There is a variety of information provided, including effort, schedule, and technical information, such as programming language, mode of development, size, and operating platform. The database has 73 descriptive fields; however, not all fields are complete for every record. At a minimum, effort and size are provided for every record. Many of the source databases NCCA used in this analysis contained more than 73 fields. Some of these fields were unique to a particular database. For instance, the Space and Missile Center (SMC) Database was the only database that had fields containing each software development review date. The various types of classification fields included in the database allow the analyst to filter data into subsets that are analogous to the program being estimated. The fields NCCA chose to include in the NCCA Raw Database have the most potential for use in a top-level software cost estimate.

The programs were developed from the early 1970s through the early 1990s. All major programming languages, including C, FORTRAN, Pascal, Ada, Assembly, JOVIAL, CMS-2, and COBOL, are represented in the NCCA Raw Database. The size of the programs range from 2.2 to 1,800 KSLOC, while the individual CSCIs range from 0.4 to 595 KSLOC. Since the program and contractor names were unknown in most of the source databases, NCCA screened the entire raw database to ensure that data points were not duplicated.

The following subsections will describe the NCCA Raw Database in detail:

- Ground Rules and Assumptions
- Data Field Definitions
- Raw Data
- Results
- Conclusions
- Future Efforts

3.2 GROUND RULES AND ASSUMPTIONS

This subsection discusses the general assumptions and ground rules followed to create the NCCA Raw Database. It is divided into three parts: Assumptions for Sizing, Assumptions for Scope of Effort and Distribution, and Other Assumptions. Some assumptions are specific to a particular source database and are described in the **Raw Data** section.

3.2.1 ASSUMPTIONS FOR SIZING

Counting SLOC can be a source of great ambiguity. It is highly probable that different people can view the same source code and use the same definition for SLOC, but count the source code very differently. The first set of assumptions addresses this problem.

- **NCCA assumed the definition of SLOC was consistent throughout a particular source database. In other words, if the author of the database stated that SLOC were measured using Delivered Source Instructions (DSI), then NCCA assumed that all the data points within that database were, in fact, expressed as DSI.**

It is important to know how the SLOC were counted so that any productivity or effort estimating relationships developed will be valid. There are two main categories of code counting conventions: physical and logical. Counting physical SLOC is accomplished by tallying the number of carriage returns in the source document. Logical SLOC are counted by tallying logical units (for example, an IF-THEN-ELSE statement is considered one logical unit).

The impact of counting convention cannot be overstated. An Institute for Defense Analyses (IDA) study of four experimental FORTRAN programs, reference [1], found that on average, physical SLOC produce a code size that is about 20 percent higher than counting the same code using a logical SLOC definition. NASA's Software Engineering Laboratory (SEL), reference [2], also found wide differences between physical and logical code counts. They found that a FORTRAN program's ratio of physical lines to logical statements ranged from 2.5 to 5 due to variations in the number of comments. Likewise, reference [2] also stated that Ada programs exhibited a similar ratio of 2.5 to 6 physical lines per logical statement.

Counting logical SLOC can be difficult since the definition of a logical unit is open to interpretation. In Ada, counting logical SLOC is easier because non-commented statements are terminated with a semicolon. Figure 3-1 shows an example of Ada source code. The right column is the actual source code. The left column shows two possible ways to count the SLOC. "1 P" indicates one physical line, "1 L" indicates one logical line, "1 C" means one comment line, and "1 B" means one blank line. The "↵" character represents a carriage return.

In the following example, there are seven physical SLOC (five are non-comment and non-blank) and four logical SLOC (and terminal semicolons).⁵ This counting result is subject to interpretation, and therefore, different results are possible. Reference [3] provides a more thorough discussion about counting SLOC. It also provides a sample checklist to help better define how SLOC are counted.

⁵The embedded semicolons in the third statement's literal string are not counted.

| Type | Source Line |
|---|---|
| 1 P, 1 L | textio.putline ("Is this a SLOC?"); ⇐ |
| 1 P | if (x=5) ⇐ |
| 1 P | then textio.putline ("How about this; one;?") ⇐ |
| 1 P, 1 L | end if; ⇐ |
| 1 P, 2 L | x:=1; y:=2; ⇐ |
| 1 P, 1 B | ⇐ |
| 1 P, 1 C | -- This is a comment ⇐ |
| P = Physical, L = Logical, B = Blank, C = Comment | |

Figure 3-1: Source Code Example

- Due to the subjectivity involved, NCCA made no attempt to define or map the source database's SLOC counting convention into a physical or logical category unless a SLOC definition was provided in the source database's documentation. Some of the embedded programs included both the terminal semicolon count and the physical SLOC count (non-comment, non-blank). When both counts were available, NCCA used the terminal semicolon count.

Not only is knowing the amount of source code necessary, but knowing the "condition" of the code is also important. NCCA uses the term condition to describe the composition of the source code (i.e., %new and %reused).

- NCCA made no assumption about the amount of new SLOC a program or CSCI contained. In other words, when the source database did not provide this level of detail, only the total SLOC was shown in the NCCA Raw Database.

The amount of higher order language (HOL) a program or CSCI contains is also an important factor to consider.

- All programming languages, except Assembly language, are defined as HOLs. If a program contained both Assembly and HOL, but the code condition (i.e., %new and %reused) for each language was unknown, then NCCA assumed the distribution for both HOL and Assembly code was the same as that for the total program.

For example, the non-shaded portions of Table 3-1 show how the source databases would typically provide their SLOC information. The numbers in the shaded boxes were not provided by the source databases, but were derived using the above assumption.

| | SIZE (KSLOC) | NEW KSLOC | REUSED KSLOC |
|------------------|--------------|-----------|--------------|
| Total Program | 1000 | 500 | 500 |
| HOL Portion | 800 | 400 | 400 |
| Assembly Portion | 200 | 100 | 100 |

Table 3-1: Allocation of New HOL and New Assembly SLOC

This assumption may not be accurate for some data points. It is possible that an even greater percentage of the new SLOC in the above example is in HOL.⁶ Analysts should ask for the new and reused SLOC by language (i.e., try to avoid having to derive values).

Total SLOC for each program in the NCCA Raw Database represents the sum of their sub-components (i.e., new, modified, verbatim, etc.). In the raw database, NCCA made no effort to convert the raw sum into equivalent new source lines of code (ESLOC).⁷

3.2.2 ASSUMPTIONS FOR SCOPE AND DISTRIBUTION OF EFFORT

When using historical software effort data, it is important to consider the level of requirements under which the software was developed. A major program may have several software development efforts spanning different acquisition phases. For example, typical acquisition strategies for new aircraft programs require a “fly-off”⁸ between at least two competing contractors. This typically requires development of prototypes and associated software during the Program Definition and Risk Reduction (PDRR) Phase, formerly known as the Demonstration and Validation (DEM/VAL) Phase. After a competitive selection process, one contractor's design is chosen for further development. Final development takes place during the Engineering and Manufacturing Development (EMD) Phase, where software development occurs once again for the deployable software. The contractor may be able to reuse code from the PDRR Phase.

The non-deployable software developed in the above example for the PDRR Phase may not undergo the same level of documentation, testing, or review as software developed in the EMD Phase for deployment. As a result, using historical PDRR data points to estimate effort in the EMD Phase may not be appropriate without further adjustment.

- **Historical software development programs in the NCCA Raw Database represent deployed software developed in the EMD Phase. Additionally, all data included in the NCCA Raw Database came from completed efforts unless otherwise indicated in the Note Fields (#69 through #71) of the database.**

Another potentially related software problem is that some of the effort may have been performed in a prior acquisition development phase. For example, it is possible that one of the software development phases, Software Requirements Analysis, took place during PDRR. This section will discuss the software development phases in greater detail later. If this occurred, and if separate contracts were signed for the PDRR and EMD Phases, then the total cost and effort

⁶ DoD program managers are strongly encouraged to minimize the new development of software in Assembly language. Assembly code is a low-level language that has processor specific commands and conventions. While very powerful, Assembly language is difficult to rehost on new or different hardware platforms (see page 3-9 for a discussion of rehosting), and it is harder to maintain. Analysts should question the accuracy of any modern data point that displays a significant amount of new Assembly.

⁷ The process (described by Boehm [5]) of converting raw SLOC into ESLOC relies on an assumed distribution of effort. (Reference [5] uses the 40/30/30 rule: 40 percent design modified, 30 percent code modified, 30 percent of integration required for modified software.) This distribution is not supported by underlying data, therefore, the analyst cannot develop an uncertainty range around the converted SLOC. NCCA develops “default” conversion factors in Sections 4, 5 and 6. These factors will have a variance associated with them to enable the analyst to perform a more accurate risk/uncertainty analysis. As more data becomes available, the conversion factors will be updated.

⁸ A competitive development between two or more contractors which produces prototypes. This culminates in a “fly-off” where each competing design is demonstrated to the user (i.e., an actual flight demonstration).

for the deployed software would be under-reported unless a portion of the PDRR costs were added to the EMD contract. Figure 3-2 depicts the problem.

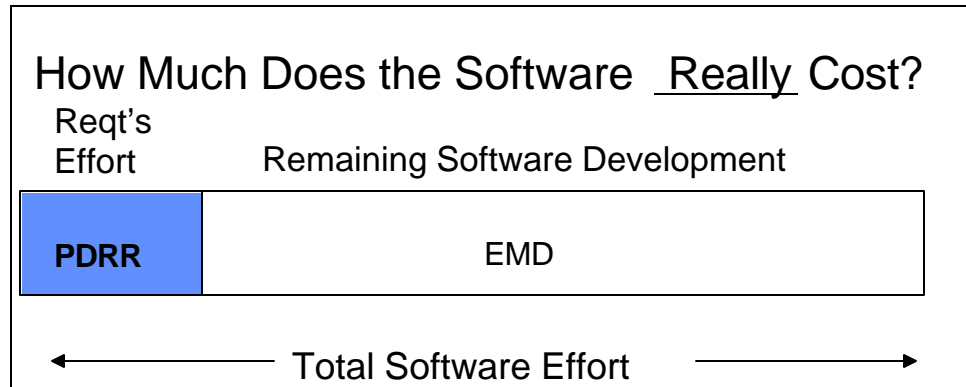


Figure 3-2: Example of Software Development Across the Acquisition Phases

- Hence, NCCA assumed all software development phases from the SDR through the FQT occurred in the EMD Phase. Furthermore, review dates are assumed to be completion dates. NCCA made no assumption about what part of the month a program started or finished.

3.2.3 OTHER ASSUMPTIONS

Simulation code is typically developed to test the operational software and is not usually delivered to the user. Therefore, it does not usually undergo the same level of rigor or documentation as operational software.

- Hence, all simulation systems were classified as ground-support systems, which are typically low in complexity. This assumes that, despite the particular mission of the software, simulation systems are inherently similar and do not suffer the same type of physical constraints as actual mission systems.

3.3 DATA FIELD DEFINITIONS

A record in the NCCA Raw Database contains 73 data fields. These fields describe various attributes of a software program. However, not all fields are complete for each record. Most records contain the following general information:

- 1) Program name, if known
- 2) Platform
- 3) Program- or CSCI-level
- 4) Size (SLOC)
- 5) Programming language
- 6) Effort expended (man-months)
- 7) Duration
- 8) Acquisition phase
- 9) SLOC counting convention

- 10) Software development phases included in the reported effort
- 11) Contractor name, if known
- 12) Notes

These 12 fields were commonly used in all of the databases that NCCA referenced and were more objective than many other fields available. In contrast, some of the fields in the NCCA Raw Database were specific to the original source database. For example, the MITRE Non-Ada Database separated the development effort by phase, while the other source databases provided total effort only.

Below is a list of field descriptions. More detailed definitions of the key fields will be provided in Section 4 - **Effort Analysis: Significant Drivers**. The terms in **bold** are field names as they appear in the NCCA Raw Database. The first 13 fields provide programmatic and classification information:

- 1) **Rec**: The key code name NCCA assigned to the program or CSCI.
- 2) **Program**: The name (if known) of the overall program (e.g., BSY-1, SLQ-32, etc.). **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Database.** See Appendix B for this information.
- 3) **Program CSCI**: The name (if known) of the CSCI in the program. "Total" is the sum of all the CSCIs within a program. **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Database.** See Appendix B for this information.
- 4) **Platform**: The major target environment of the operating software, including the following designations:
 - Ship**: Ship or submarine-based system
 - Ground**: Ground-based system, including simulation systems
 - Air**: Manned or unmanned aircraft or missile system
 - UNMNDSP**: Unmanned space system
- 5) **CSCI?**: This field contains "Y" if the record is CSCI-level and "N" if it is program-level. No data was collected below the CSCI-level (i.e., at the Computer Software Component (CSC) or Computer Software Unit (CSU) - level).
- 6) **SW Class**: A classification of the software's top-level function. Software is typically classified into three categories: application, support, and system. System code is the most difficult to develop, while support code is the least difficult. Refer to Section 4 - **Effort Analysis: Significant Drivers** for further definitions and examples.
- 7) **Status**: A classification of the software's mission as it pertained to the end user.

The status classifications were operational and non-operational as defined below.⁹

OP: Operational software (i.e., actual mission software that was delivered to the user).

NOP: Non-operational software, as follows:

PGS: The program generation support software used to develop the operational software, but not necessarily delivered to the user.

SIM/STIM: The software developed to test operational software, but not necessarily delivered to the user (e.g., software utilized to generate artificial sonar signatures to test a sonar system).

8) **Mission:** The specific function the software performs. For a program with multiple CSCIs, this field contains the same value for each CSCI. The NCCA Raw Database includes the following missions:

C³: Command, control, and communications (C³). Information systems that gather, control, process, and distribute strategic, tactical, intelligence, and message data (e.g., MILSTAR and GPS).

RADAR: Systems that use microwave energy to detect, determine range, and track ground or airborne targets as defined in reference [4]. Includes all radar on ships, aircraft, and ground systems (e.g., SPY-1).

SIM: Simulation systems (e.g., F-18 trainers).

EW: Electronic warfare systems, such as jamming and countermeasure systems (e.g., ASPJ).

SONAR: Ship systems, similar to radar, which use sound waves instead of microwaves (e.g., BQQ-5).

ASW: Anti-submarine warfare systems. A collection of major sub-systems that detect and/or neutralize enemy submarines (e.g., BSY-2).

TORP: Air-, ship-, or submarine-launched weapon systems that move through water to destroy ships or submarines (e.g., MK48).

MINE: Explosive devices that deter both personnel and enemy vehicles from entering a protected area. Mine systems include mine countermeasures (e.g., SLQ-32).

MISSILE: Ground-, sea- or air-launched, self-powered weapon system (e.g., AIM-9X and Standard Missile).

⁹None of the source databases had a field to track the operational status of the code, but NCCA plans to collect this type of information for future analysis. NCCA attempted to classify the operational status of the SLOC in the current version of the database, but subsequent analysis of the data revealed either non-significant or illogical relationships. See Section 4 – Effort Analysis: Significant Drivers for more on this analysis. Therefore, NCCA will clear this field in later versions of the database (i.e., all records in the database will have a blank Field #7) until more accurate data is obtained.

MIS: Management Information Systems. Non-weapon system, such as financial systems, inventory systems, decision support systems, etc. (e.g., NALCOMIS and JCALS).

9) **Major Function:** Some data sources supplied more information on the specific function of the CSCI (e.g., display or message processing or executive control). Unlike the Mission field, this field may vary at the CSCI-level within the same program.

10) **Lang1:** The name of the primary (greater than 50 percent of the total SLOC) programming language used to code the software.

11) **Lang2:** The name of the programming language that constitutes the next highest percentage of the total SLOC.

12) **CSCI Count:** For program-level data, the total number of CSCIs in the program. For CSCI-level data, the count equals one. For program-level data the count can also equal one (i.e., the program consisted of only one CSCI).

13) **HOL:** The percentage of total SLOC written in an HOL. If this field contained a value, it was between zero and one.

The next seven fields refer to software size. Sizing within the NCCA Raw Database was expressed in SLOC. Documented sources seemed to follow Boehm's counting convention [5] of including only delivered code.¹⁰ Fields #15 through #19, expressed as percentages, further defined total SLOC. If they contained a value, it was greater than or equal to zero, but less than or equal to one.

14) **Total:** This is the total raw sum of new and reused (modified, verbatim, rehosted, translated) SLOC.

15) **New:** The percentage of the total SLOC that is new. New was defined as "freshly made and unused."

16) **Mod:** The percentage of the total SLOC that required some amount of redesigning, recoding, and retesting. The effort to modify code is usually less than the effort to create new code.

17) **Verbatim:** The percentage of the total SLOC that was used "as-is" with no redesigning or recoding. Note that this code may or may not need retesting at the CSU-level.

18) **Rehosted:** The percentage of the total SLOC originally written for one source architecture, but moved to another (sometimes different) architecture. For example, taking Lotus 1-2-3 source code from the Intel 486 platform and compiling it to run on the Motorola 68000 platform is a rehosting effort. A second interpretation of rehosted code is transferring code from one operating system to another. An example of this type of rehosting effort is to

¹⁰Non-delivered code includes "deleted" SLOC. This count is important, but is usually not captured. Deleted code indicates how much rework is performed if the program is entirely new. SLOC may also be deleted when programs are reused or "re-engineered." While deleting code in and of itself probably is not too difficult, there are times when problems occur, resulting in additional and unexpected effort.

convert Lotus 1-2-3 source code written for Disk Operating System (DOS) and operate it under the OS/2 operating system. Consequently, it is possible to find software programs where the rehoused code not only represents changing architecture, but also represents changing operating systems.¹¹

19) **Translated:** The percentage of the total SLOC that was converted from one source language to another. Converting a program from C to Ada is an example of translated code. Typically, using an automatic translator which converts one programming language to another can save much effort.

20) **Comments:** The total number of comment SLOC.

Fields #21 and #22 (Name and Count) identify how the SLOC were counted:

21) **Name:** The source database classification for SLOC.

DSI: Delivered Source Instructions. As discussed in reference [5], DSI include delivered executable SLOC, data declarations, job control language (JCL) statements, and INCLUDE files (counted once). DSI exclude comments, prefaces, file boundary statements, COTS software that is not modified, non-delivered support software, and non-delivered test software. Note: NCCA defined DSI as logical SLOC. See reference [3].

CR: Carriage Return. All lines are counted regardless of programming style, including comments and blanks. May or may not include non-deliverable test software. NCCA classified this as physical SLOC. See references [1] and [2].

TSC: Terminal semicolons.¹² This terminology is usually associated with Ada and C++ programs. Includes all statements that terminate with a semicolon. NCCA classified this as logical SLOC. (See reference [1].) Counting with terminal semicolons is the more objective method for counting logical SLOC.

SLOC: Software Architecture Sizing & Estimating Tool (SASET) defines SLOC as source lines of code that “consist of all executable statements, plus inputs/outputs, format statements, data declaration statements, deliverable JCL statements, and procedure-oriented language statements. SLOC does not include statement continuations, database contents, CONTINUE statements, or program comments” [6].

SMC [7] defined SLOC as “. . . a single instruction, not necessarily a physical line. Comments are not counted. As an example in Ada, source SLOC are counted by the number of [terminal] semicolons” and include the following statements:

¹¹Legacy code that is written in Assembly language is difficult to rehost.

¹²The terminal semicolon count and other logical counting schemes are preferred because they produce a size parameter that is less sensitive to coding style. Since programming styles influence physical SLOC counting conventions, programmer productivity could be artificially high (when productivity is defined as physical SLOC per hour) merely because compound constructs are broken into separate lines. For instance, if an IF-THEN statement is split into more than one line, it would count for more than one physical SLOC. In Ada, executable and declarative statements are terminated with a semicolon. This means that in the previous example, a multi-line IF-THEN statement counts as one statement. Care must still be taken to ensure that if an automated code counter is used, it does not count semicolons embedded in comments or strings. See reference [3] for a discussion of the issues surrounding counting SLOC.

Control (DO While, DO Until, GOTO)
Mathematical ($i=a**b=c$)
Conditional (IF-THEN-ELSE)
Deliverable JCL statements
Data declaration statements
Data typing statements and EQUIVALENCE statements
Input/Output format statements

but exclude

Comments
Blank lines
BEGIN statements from BEGIN-END pairs
Non-delivered programmer DEBUG statements
Continuations of format statements
Machine- or library-generated data statements

Even though the SASET and SMC definitions of SLOC were slightly different, NCCA classified both of these as logical SLOC. The Software Engineering Institute (SEI) [3] defined SLOC as physical SLOC. Other source databases used the generic term SLOC, but did not provide definitions.

22) **Count:** Based on the previous definitions, this field contains NCCA's mapping of the source database's SLOC classification (Field #21).

CP: Commented Physical SLOC

P: Non-commented Physical SLOC

L: Non-commented Logical SLOC

?: Not enough information for an assessment

The next seven fields provide development and process information:

23) **Mode:** The development environment of the software program. Mode designations are embedded, semi-detached, or organic.¹³ Refer to Section 4 - **Effort Analysis: Significant Drivers** for further details.

24) **Period:** The acquisition phase of contract performance. This field contains either CED (Concept Exploration and Definition), PDRR (previously known as DEM/VAL), EMD or a blank, if unknown. NCCA has not discovered any data points that span more than one acquisition phase.

¹³ The MITRE Non-Ada Database [6] also included a fourth mode called firmware (FW). The Mitre Non-Ada Database (DB #1), the Mitre Ada Database (DB #2), the REVIC Recalibration Database (DB #7), and certain programs from the Navy Internal Database (DB #5) provided the development mode. For the remaining databases, NCCA subjectively mapped programs into the applicable mode designations, if enough information was provided. For an example, refer to the SMC software database discussion on page 3-19.

25) **MM.eq.152:** Contains either “Y” or “N.” If “Y” appears in this field, then either the source database recorded the effort in hours, used a 152-hour man-month, or used a different hour per man-month value and NCCA normalized the reported effort to a 152-hour man-month.¹⁴ If “N” appears in this field, the source database did not provide the hours per man-month.

26) **Hrs/MM:** The actual hours per man-month recorded by the source database. Values significantly less than 152 hours per man-month may indicate that the schedule was extended and values significantly greater than 152 hours per man-month may be an indication that the schedule was compressed. Most schedule estimation models are based upon a historical database of actual schedule and actual effort. When the hours per man-month are increased, the corresponding schedule decreases. The values in this field ranged from 150 to 176 hours per man-month.

27) **Method:** The software development method (process) followed. The types of methods are waterfall, incremental, spiral, and evolutionary. Refer to the NCCA Software Glossary and Primer or reference [5] for further details.

28) **Peak:** The highest number of persons used during the development effort at one time.

29) **Average:** The average size of the program staff (number of persons) during development. This was calculated by dividing the total effort by the total elapsed development time. The values for effort and time should be of the same scope (i.e., if the effort scope is SDR through FQT, then schedule scope should also be SDR through FQT).

The next seven fields describe the amount and scope of effort:

30) **TotalMM:** The total effort (man-months) expended to develop the software. If the source database provided a breakdown, total effort was decomposed into the following software development phases:

31) **Reqmnts:** The total effort expended during the requirements phase. Activities include a basic draft of the software documentation, including the SDP, Software Requirements Specification (SRS), and Software Test Plan (STP). The Interface Control Document is also finalized during this phase. Most of the source databases either did not include this effort or did not state that the effort was included elsewhere. Requirements are sometimes generated during the PDRR Phase of the program, and therefore, may not be captured on an EMD contract.

32) **EDT (Design Total):** The total effort expended in the design phase. This is the sum of Fields #33 and #34 (Preliminary and Detailed Design Effort), as defined below:

33) **EPD (Preliminary Design):** All effort expended between the Software Specification Review (SSR) and Preliminary Design Review (PDR). Activities include finalizing the system

¹⁴For example, if the source database reported total effort as 100 man-months and stated that 1 man-month = 173 hours, then NCCA re-calculated the effort and recorded it as $173/152 \times 100 = 114$ man-months. Most of the source databases recorded their effort using 152 hours per man-month; therefore, NCCA normalized man-hours to this standard.

requirements specifications (A spec), finalizing the CSCI test plans, and finalizing the database requirements. Functional design specifications for each CSCI, including identification, sizing, and language, are generated.

34) **EDD (Detailed Design):** All effort expended between the PDR and the Critical Design Review (CDR). Activities include preliminary test procedures, detailed flow charts for each CSCI, database specifications, and a final requirements traceability matrix.

35) **ECUT (Code and Unit Test):** All effort expended from the CDR to the Test Readiness Review (TRR), including testing of individual software units and informal testing.

36) **EIT (Integration and Test):** The effort to successively integrate and test all CSCs. This includes the FQT of each CSCI, but does not cover the CSCI-to-CSCI integration nor the integration and testing of the hardware with the software system [8].

In addition to providing the amount of effort by phase, the NCCA Raw Database has nine fields describing what is and is not included in the reported effort. These nine binary fields contain a zero, one, or blank (if unknown). These fields used ones and zeros so that they could be used as dummy variables in later regression analyses.

37) **REQ:** Did the reported total effort include effort for the Software Requirements Phase? Yes = 1, No = 0, Blank cell = unknown.

38) **PD:** Did the reported total effort include effort for the Preliminary Design Phase? Yes = 1, No = 0, Blank cell = unknown.

39) **DD:** Did the reported total effort include effort from the Detailed Design Phase? Yes = 1, No = 0, Blank cell = unknown.

40) **CUT:** Did the reported total effort include effort for the Code and Unit Test Phase? Yes = 1, No = 0, Blank cell = unknown.

41) **CSC TST:** Did the reported total effort include effort for the CSC Testing Phase? Yes = 1, No = 0, Blank cell = unknown.

42) **CSCI TST:** Did the reported total effort include effort for the CSCI Testing Phase? Yes = 1, No = 0, Blank cell = unknown.

43) **SIT:** Did the reported total effort include effort for the System Integration and Test Phase? Yes = 1, No = 0, Blank cell = unknown.

44) **OTE:** Did the reported total effort include effort for the Operational Test and Evaluation Phase? Yes = 1, No = 0, Blank cell = unknown.

45) **Other:** Was another phase other than those mentioned above included? Yes = 1, No = 0, Blank cell = unknown.

The next three fields specify the program's software schedule. The detailed reviews are defined in Fields #54 through #64.

46) **Start:** The completion date of the SDR. If this date is unknown, the field is blank.

47) **Finish:** The completion date of the FQT. If this date is unknown, the field is blank.

48) **Total Months:** The elapsed time from start to finish. Units are calendar months. If either the start or finish were unknown, then this field was blank. There are some programs that did not have start or finish dates, but provided the elapsed time. This elapsed time was included only if the documentation indicated that the start and finish were defined as SDR and FQT, respectively.

The next five fields capture elapsed times for specific phases in the software development effort. Elapsed time is in calendar months.

49) **TTD:** The total elapsed time for preliminary and detailed design combined. This field is the sum of Fields #50 and #51 below.

50) **TPD:** The elapsed time for the Preliminary Design Phase.

51) **TDD:** The elapsed time for the Detailed Design Phase.

52) **TCUT:** The elapsed time for the Code and Unit Test Phase.

53) **TIT:** The elapsed time for the Software Integration and Test Phase, but does not include the System Integration and Test Phase.

The next eleven fields, as shown in Figure 3-3 and defined below, provide completion dates for the various software reviews. These reviews (and their definitions) come from DoD-STD-2167A. The adoption of MIL-STD-498, J-STD-016, ISO-12207 may pose trouble for collecting and analyzing this type of data in the future. MIL-STD-498 shifts the focus from “formal” reviews to “informal” reviews. Thus, many of the following reviews may not occur:

54) **SRR:** System Requirements Review provides insight into the developer’s plan for the system configuration.

55) **SDR:** System Design Review provides insight into the overall system requirements as a basis for establishing the system specification’s functional baseline.

56) **SSR:** Software Specification Review is a formal review of a CSCI’s requirements per the software specifications.

57) **PDR:** Preliminary Design Review provides insight into the developer’s progress and the correctness of the software components’ design.

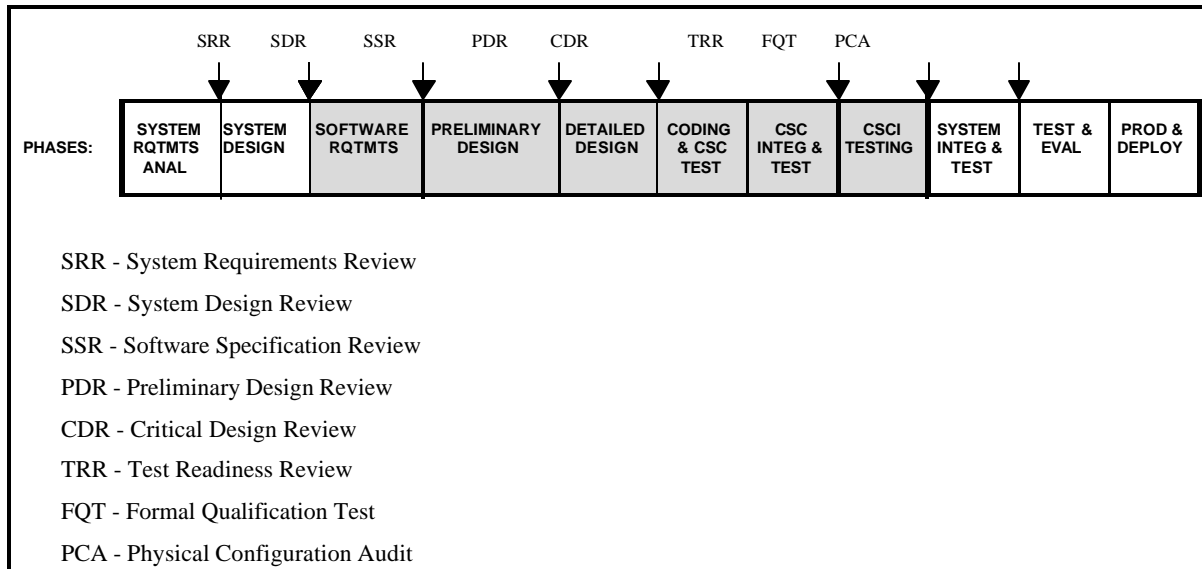


Figure 3-3: Software Development Phases and Reviews

58) **CDR:** Critical Design Review is a review that determines if the software design satisfies the requirements of the system and software specifications.

59) **PQT:** Preliminary Qualification Test is a phase not recognized in DoD-STD-2167A, but it was a field identified in one of the source databases. This review occurs after the coding and CSC testing has been completed.

60) **TRR:** Test Readiness Review verifies that the developer has performed their own testing and has the resources, plans, and procedures to formally demonstrate to the government that the software works as an entity.

61) **FCA:** Functional Configuration Audit is performed by the government to determine if CSCIs perform in accordance with their respective requirements and interface specifications by examining the test and reviewing the operational and support documentation.

62) **PCA:** Physical Configuration Audit is the formal technical examination of the as-built software product against its design.

63) **FQT:** Formal Qualification Test is the formal testing of the CSCI per the government approved test plans and procedures to verify that the CSCI fulfills the requirements of the SRS and to provide the basis for CSCI acceptance by the government.

64) **OTEVAL:** Operational Test and Evaluation

More detailed definitions of these reviews can be found in the NCCA Software Glossary.

The next five fields provide programmatic information:

65) **STD:** The military standard the software program was required to follow. Possible values are "2167," "483/490," "1521," "1679," and "498." MIL-STD-498 (498) replaced

DoD-STD-2167A in 1993; however, the NCCA Raw Database does not yet contain any programs developed under 498.¹⁵

66) **Contractor:** The name of the prime contractor performing the software development. **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Database.** See Appendix B for this information.

67) **Source:** The source of information (other published software databases or cost reports) for each data point in the NCCA Raw Database. **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Database.** See Appendix B for this information.

68) **DB Code:** The code number NCCA assigned to each source database. It is included in both the raw and the sanitized version of the NCCA Raw Database. The index maps to the following databases:

- 1) MITRE Non-Ada Database
- 2) MITRE Ada Database
- 3) SMC Database
- 4) NASA SEL Database
- 5) Navy Internal Data Sources
- 6) Silver SASET Calibration Study Database
- 7) Revised Intermediate Constructive Cost Model (REVIC) Recalibration Study Database
- 8) IIT Research Institute (IITRI) Report Database

69) through 71) **Notes 1 through 3:** These fields inform the analyst of any specific assumptions or peculiarities about the data point. They also describe potential problems with the data (e.g., the source did not indicate the hours per man-month associated with the effort). **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Database.**

Finally, the last two fields are calculations generated by NCCA:

72) **Prod1:** Productivity expressed as hours per SLOC based on the following formula:

$$\text{Hours/SLOC} = \frac{\text{Total MM} * 152 \text{ Hours/MM}}{\text{Total SLOC}}$$

This field is blank if Field #25 (MM.eq.152) equals "N."

73) **Prod2:** Productivity expressed as hours per new SLOC. This metric is important because new SLOC tend to drive the effort. (See Sections 5 and 6 for a more detailed explanation)

¹⁵NCCA is not currently collecting any software development efforts under commercial standards. However, MIL-STD-498 expired in 1996, at which time, DoD intended to adopt the International Standards Organization (ISO) standard for software development. Refer to the NCCA Issue Paper "MIL-STD-498 versus DoD-STD-2167A" for a discussion of this topic.

$$\text{Hours/SLOC} = \frac{\text{Total MM} * 152 \text{ Hours/MM}}{\text{Total SLOC} * \text{New SLOC}}$$

This field is blank if Field #25 (MM.eq.152) equals "N" or Field #15 (New) is blank or zero.

3.4 RAW DATA

This section describes the data sources, the type of data collected, and any general problems, comments, and adjustments or normalizations that were made to the raw data before entering it into the NCCA Raw Database.

Most of the NCCA Raw Database originated from either published reports or externally developed software databases. NCCA also collected some Navy internal data from previous Independent Cost Estimates (ICEs) and other analytical efforts. A total of eight source databases were either used directly or as a reference for clarification:

- 1) MITRE Non-Ada Database
- 2) MITRE Ada Database
- 3) SMC Database
- 4) NASA SEL Database
- 5) Navy Internal Data Sources
- 6) Silver SASET Calibration Study Database
- 7) REVIC Recalibration Study Database
- 8) IITRI Report Database

3.4.1 MITRE NON-ADA DATABASE

Published in a 1987 MITRE report [8], this database contained over 110 CSCIs from 26 programs. Nineteen of the 26 programs came from the Air Force Electronic Systems Center (ESC). Twenty-three of the 26 programs were complete. The CSCI sizes ranged from 0.4 to 492 KDSI, and the program sizes ranged from 8 to 1,113 KDSI. The data originated from standard cost forms called M-forms. The forms covered five different areas:

- 1) Program summary
- 2) Development and target computer
- 3) Computer Program Configuration Item (CPCI) summary
- 4) Resource expenditure
- 5) CPCI function and sizing

Data sources for the MITRE internal and external contractor supplied data included technical reports, working papers, interviews, PDR and CDR briefing charts, and software requirement and product specification charts (B5 and C5 system engineering specifications). MITRE attempted to use multiple sources for each program to verify that the data collected was correct. Their review resulted in the modification of some of the programs' initial SLOC counts and Effort Adjustment Factor¹⁶ (EAF) multipliers.

¹⁶Special environment variables originally developed in the Constructive Cost Model (COCOMO) [5].

The MITRE Non-Ada Database defines Delivered SLOC (DSLOC) as new code plus adapted DSI. Some of the data points that had adapted DSI included the percent of adapted DSI that was redesigned, recoded, reintegrated, and retested. The NCCA Raw Database classified any data point that was zero percent redesigned and recoded, but was reintegrated and retested, as verbatim SLOC. The remaining adapted DSI (where the percentage of redesign and recoding were not equal to zero, or any data points that did not provide the percentage of redesign, recoding, and retesting information) were classified as modified SLOC.

The MITRE Non-Ada Database counted SLOC using Boehm's DSI definition (see page 3-9). MITRE also counted and reported the comment SLOC. NCCA included this information in the NCCA Raw Database in Field #20 (Comments). MITRE's SLOC were mapped as logical SLOC.

The programs for which MITRE collected data were written primarily in Assembly, FORTRAN, or Jovial, and were developed from the 1970s to 1980s. The data collected represents development efforts following MIL-STD-1521A or MIL-STD-1521B. However, since the documentation did not specify which programs were developed under 1521A vice 1521B, the development standard was not entered into the NCCA Raw Database. MIL-STD-1521A did not define the SSR, but 1521B did.

Effort data in the MITRE Non-Ada Database reflected all direct labor costs to develop software, including management, designing, programming, testing, and data collection. It did not include system requirements analysis, implementation (installation, conversion, or training), and maintenance [8]. All effort was normalized to a 152-hour man-month and spanned SDR through FQT.

NCCA deleted several of the MITRE Non-Ada data points for the following reasons:

- 1) Data points #8, #10, #20: Programs were either halted or were incomplete. Program #10 (Peace Shield) was incomplete in the MITRE Non-Ada Database, but was subsequently captured, when completed, in the SMC Database.
- 2) Data point #2: Effort was shown for all phases except Code and Unit Test. It was not clear whether this effort was allocated to the other phases or was not available, hence not included.
- 3) Data point #18: Total effort was reported for the entire program, but the CSCIs for the program did not sum to the program total. This implies that there was a mapping or allocation problem.

Fields #33 through #36 (EPD, EDD, ECUT, and EIT) were used primarily in the MITRE Non-Ada Database. The database also captured the COCOMO environment variables and various other metrics. Due to the subjectivity of these variables, they were not included in the NCCA Raw Database. Additionally, several of the programs provided effort by development phase. MITRE provided effort for Preliminary Design, Detailed Design, Code and Unit Test, and Integration and Test. However, software requirements were not separately identified; thus, NCCA does not recommend using the MITRE Non-Ada Database for phase-specific effort estimation or for top-level distribution of effort.

Seventy-two data points were collected from the MITRE Non-Ada Database. Overall, there is enough high quality data to perform a top-level normalized analysis using this database, but there are disadvantages:

- 1) The database contains no Ada programs.
- 2) The programs are old.
- 3) Some programs use significant amounts of Assembly language.
- 4) Program and contractor names are unknown in most cases.

3.4.2 MITRE ADA DATABASE

Published in a 1992 MITRE Report [9], this database contained information from 18 Ada programs with a total of over 50 CSCIs. The database included software data for both weapon system (e.g., avionics) and MIS applications. CSCI sizes ranged from 0.7 to 228 KDSI, and programs ranged from 2.3 to 340 KDSI. This database tallied SLOC by counting terminal semicolons. SLOC were divided into new, modified, and “lifted.” NCCA mapped lifted SLOC into verbatim SLOC in the NCCA Raw Database.

The reported effort followed the same Boehm definitions [5] and phases that were described in the MITRE Non-Ada Database, with the exception that the effort in the MITRE Ada Database covered the period from SSR through FQT. Consequently, software requirements analysis was not included in the total effort.

The semi-detached programs in the database came from a 1987 study which did not count semicolons, but instead used Delivered SLOC. A 1.33 DSLOC to one semicolon factor was used in the study to convert size data given in DSLOC to semicolons. This was one source of variation (or error) in the database. No information was provided about the variance surrounding the conversion factor. To avoid introducing subjectivity, NCCA converted the semi-detached programs to their original SLOC (DSLOC) count and defined the SLOC as physical SLOC.

The MITRE Ada Database included a field called “PDL Lines.” PDL stands for Program Design Language. PDL Lines are English-like statements (also referred to as “pseudo-code”) that aid the software engineer during the Preliminary Design Phase. Some of the programs used Ada as the PDL. NCCA defined PDL Lines as non-delivered code. Since other programs tracked only delivered code, PDL Lines were not included in the total SLOC in the NCCA Raw Database. The MITRE Ada Database Data Input Form clearly asked if PDL Lines were included or excluded from the total SLOC counts, so accurate mapping was possible. Although PDL Lines were not included in the NCCA Raw Database, they may be a useful metric in the future as an indicator of the scope of effort in the design phase.

Thirty data points were captured from the MITRE Ada Database. However, several data points from the MITRE Ada Database were deleted for the following reasons:

- 1) Data point #8: Duplicate data point (AFATDS) from another database.
- 2) Programs “?5” and “?6”: The CSCIs’ names (“ENGLISH,” “FRENCH,” “DICTIONARY”) imply that they were commercial programs vice defense related efforts.

This database was promising because of the number of Ada programs. However, the database does not capture the Effort Software Requirements Phase (i.e., effort did not start at SDR) and, therefore, this database is not recommended for estimating the full scope of effort (SDR through FQT) for software programs. If software requirements analysis does not need to be estimated, the MITRE Ada Database is useful for top-level non-normalized productivity factors and regressions. Additionally, schedule estimation with this database would require adjustments, since schedule was also typically captured from SDR through FQT. Since the effort was collected starting at the SSR, an entire phase (Software Requirements Analysis) of effort was missing. The implication is that an analyst would have to either construct regressions to estimate a full software development schedule using an independent variable that does not reflect the full software development effort or artificially inflate the effort to reflect the full phasing required.

3.4.3 AIR FORCE SPACE AND MISSILE SYSTEMS CENTER (SMC) SOFTWARE DATABASE

The Air Force SMC Software Database is a 2,600 record database maintained and updated by Management Consulting and Research, Inc. (MCR) under contract to the Air Force SMC. The data was originally contained in the Space Systems Cost Analysis Group (SSCAG) Software Database and has since been expanded. The database is in a Windows format that can be queried and exported to a spreadsheet.

The SMC Database has extensive information for each record and covers program-level down to CSU-level information. However, many of the records' fields are blank. NCCA extracted as many well defined data points as possible from this database. A data point was considered well defined if the following information was available: size, effort, and hours per man-month.

The SMC Database's SLOC were mapped as logical SLOC and included information on both new and reused SLOC. In the SMC Database, reused code may have contained some information about the percent of code that was redesigned, modified, or retested. Similar to the MITRE Non-Ada Database, any record in the database that contained reused SLOC, but did not contain information about the percent redesigned, modified, or retested, was mapped into modified SLOC. If the reused code had zero percent redesign and zero percent re-coding, NCCA mapped it into verbatim SLOC. The SMC Database also contained a field to track the level of rehosting that a program experienced. The values ranged from NOMINAL to EXTRA HIGH. SMC defined VERY HIGH as a program with a major language or system change (refer to Field #19 (Translated) in the Data Field Definition section for further information). EXTRA HIGH is defined as a program having a major language and system change. If the program indicated the rehosting effort to be VERY HIGH or EXTRA HIGH, then the reused SLOC from the SMC Database was mapped into the NCCA Raw Database as rehosted code.

In addition to new and reused SLOC, the SMC Database included common SLOC. Figure 3-4 shows a graphical representation of the difference between new and common SLOC. Common SLOC are SLOC in an individual CSCI that are identical to code in other CSCIs of the same program.

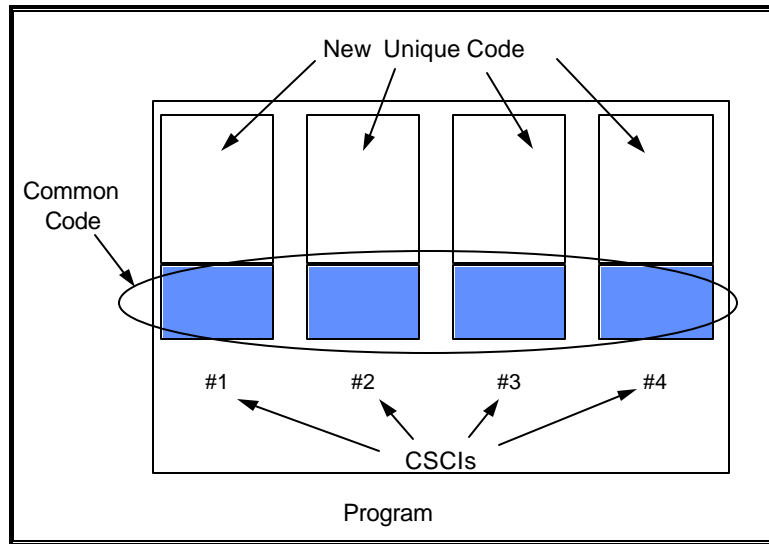


Figure 3-4: New Code versus Common Code

It was difficult to use this information because the SMC Database did not tag these CSCIs to indicate which ones belonged to the same program. At the program-level, common SLOC were counted once as new SLOC in the program count. However, at the CSCI-level, several counting approaches could have been utilized. The common SLOC could have been counted as new SLOC for the CSCI in which it was first developed, while subsequent CSCIs that used the common SLOC would have counted it as reused SLOC. A second approach to counting common SLOC at the CSCI-level would have been to prorate it into each CSCI's new SLOC count. A total of five SMC data points in the NCCA Raw Database have common SLOC. In the end, NCCA mapped common SLOC into verbatim SLOC.

A total of 38 unique data points from the SMC Database were included in the NCCA Raw Database. Some of the 38 records did not contain the hours per man-month. However, NCCA reviewed an older version of the SMC Database (the SSCAG Database) from 1989, which explained that the original raw data was expressed in hours and was converted to man-months using the 152 hours per man-month conversion factor. Thus, NCCA was able to verify that 152 hours per man-month were utilized for 35 of the 38 data points. An additional 41 data points from the same database were duplicated in the REVIC Recalibration database.

The SMC Database contained programs and CSCIs with different scopes of effort. The SMC Database used "check boxes" to indicate what software development phases were included in the reported effort. NCCA adopted a similar approach to delineate which phases were in the reported effort for the SMC Database and all other data sources (if this information was known).

The SMC Database did not contain a mode field; hence, NCCA used internally developed rules to determine the mode of development. The data points were classified as "embedded" if the architecture field indicated that it was "tightly coupled" or if the requirements volatility rating was set to at least HIGH. The data points were considered "not embedded," if the points did not satisfy these criteria (i.e., no attempt was made to discriminate between semi-detached and organic programs).

Additionally, some SMC data points indicated the percentage of total effort by contractor functional area (i.e., systems engineering, configuration management, quality assurance, data, etc.). This information was not included in the NCCA Raw Database, but may be useful in future research. The SMC Database also included a section for maintenance data. Unfortunately, there were only two programs in the database with maintenance information.

Overall, the 79 data points extracted from the SMC Database are of high quality. Weaknesses of these data points include the following:

- 1) The linkage of CSCIs to their associated programs was unknown.
- 2) The database did not provide the CSCI count for program-level data, which may be an important factor when estimating integration costs.
- 3) Program and contractor names were unknown, which reduces the users' ability to identify the best analogies possible and increases the risk of duplicating data points when combined with other databases.
- 4) Effort information, by phase, was not provided.
- 5) The database did not indicate if CSCI schedules were specific to individual CSCIs or represented the top-level program schedule.

3.4.4 NASA SOFTWARE ENGINEERING LABORATORY (SEL) DATABASE

The SEL is sponsored by the NASA Goddard Space Center in Maryland. The SEL collects and analyzes software development data to investigate the effectiveness of software engineering technologies.

References [2] and [10] provided a total of 37 data points to the NCCA Raw Database. These programs were NASA programs which were predominately ground-based, satellite-support programs. The support software programs were either attitude ground-support or telemetry-simulation software. The software applications were used to determine and predict the orbit and attitude of Earth-orbiting satellites. The data covered the mid-1970s to early 1990s. A total of 11 programs were written in Ada; the rest were written in FORTRAN. The programs ranged in size from 9.1 to 338 KSLOC.

The data in the above referenced reports is a subset of a much larger NASA SEL Database. This larger database, maintained by the Rome Air Development Center, contains extensive information on 104 programs and includes error data, detailed product characteristics, effort, growth history, change history, and program information. NCCA did not attempt to collect data from this database, since additional data collection efforts will be the focus of NCCA's future software efforts.

The SEL counted SLOC by including every carriage return in the source code. This count included comments and blank lines [10]. NCCA interpreted this as physical SLOC. However, the 1995 report [2] added statement counts as well. Statements are defined as the number of

logical statements and declarations. Recall that NCCA's first choice is to use the statement count; however, not all programs had both counts. For those programs that did not have a statement count, NCCA used the total SLOC and classified them as commented physical (CP) SLOC in the NCCA Raw Database. Programs that had statement counts were classified as logical (L) SLOC.

The SEL segregated source code into newly written, extensively modified (25 percent or more of the code was modified), slightly modified (less than 25 percent of the code was modified), and verbatim code. NCCA mapped newly written code into Field #15 (New), extensively and slightly modified code into Field #16 (Mod), and verbatim code into Field #17 (Verbatim). (See the Data Field Definitions section).

The SEL Database provided effort in man-hours by both phase (design, code and test) and WBS activity. Schedule was provided in months by phase. The effort covered from pre-program (software requirements) through clean up (system acceptance and test). The data also captured other support efforts such as upper management, librarians, technical publications, and secretarial support over the same period. Some of these efforts (upper management and secretarial support) were overhead, while others were classified as direct support (librarians and technical publications). Since these indirect efforts cannot be separated from other direct support efforts, the entire effort was included in total effort in the NCCA Raw Database. Additionally, effort was converted to man-months using the 152 hours per man-month factor. No information was given as to the actual hours per man-month (factory hours per man-month) that the staff experienced; however, this did not create a normalization problem since the original reported units of effort were in man-hours.

The SEL Database included both System Test and Acceptance (Operational) Test in the reported effort. However, since it was identified separately, NCCA was able to remove all system-level testing from the reported effort. Therefore, SEL data points in the NCCA Raw Database reflect effort from SDR to the completion of the CSCI test phase (FQT).

The SEL schedule data included preliminary design efforts. Hence, NCCA mapped the SEL Database's program start date to the completion of the SSR, Field #54 (recall from Figure 3-3 that preliminary design starts after SSR). The SEL Database included both system and acceptance testing. Therefore, the program's end date was mapped to the completion of the Operational Test and Evaluation (OTE) in the NCCA Raw Database because completion of OTE guarantees that all system and acceptance testing is complete.

Since the other databases displayed effort by phase only, the NCCA Raw Database used SEL effort data by phase versus effort by WBS task to remain consistent. However, the SEL Database did not break out the aforementioned support (direct and indirect) by phase, but instead reported total support. As such, NCCA distributed the support across the phases using the following allocation formula:¹⁷

$$E_{iADJ} = E_i + \frac{E_i}{E_{Total}} * \text{Support}$$

¹⁷NCCA assumed that support would be present in both the System and Acceptance Test Phases. Therefore, the effort for these phases was included to arrive at the proper allocation of effort to all phases (i.e., this distribution was performed before removing the System and Acceptance Tests from the NCCA Database).

where E_{iADJ} is the effort in phase i , including support; E_i is the effort in phase i , excluding support; E_{Total} is the sum of the individual phased efforts; and support is the total support effort. Therefore,

$$\text{Total } E_{iADJ} = E_{Total} + \text{Support}$$

The strengths of the SEL Database are as follows:

- 1) All data represented the same domain, and all programs fell within two categories: attitude ground-support or telemetry-simulation software. This created a better data set to analyze for changes in the development processes.
- 2) The SEL Database's productivity reflected modern processes and constant process improvement.

The weaknesses of the SEL Database are as follows:

- 1) Software developed for NASA may not have the same level of documentation requirements as software developed in DoD. Software developers at NASA are encouraged, but not required, to develop software consistent with the guidelines set forth in reference [11], which closely resemble DoD-STD-2167A.
- 2) All of the software was support software.
- 3) Only a few of the programs counted logical SLOC. The rest counted physical SLOC.

Still, the SEL data was of sufficient quality to be used for top-level normalized productivity factors and regressions.

3.4.5 NAVY INTERNAL DATA

Several sources of contractor-specific data, internal to the Navy, were included in the NCCA Raw Database. These data points were gathered by NCCA analysts in support of past ICEs and other analyses. These data points represent software developed from the 1980s to mid-1990s. Due to the business sensitive nature of the data, the source and contractor names are not available in this document. This section provides an overview of the business sensitive sources. Refer to Appendix B for specific details.

The NCCA Raw Database included a total of 127 data points, 50 at the program-level and 77 at the CSCI-level, from Navy internal data sources. The data covered a broad spectrum of Navy systems, including sonars, combat systems, trainers, electronic warfare, and C³. Programming languages included FORTRAN, Ada, C, Pascal, and Assembly. The CSCI sizes ranged from 3.7 to 342 KSLOC, and the program sizes ranged from 2.6 to 1,421 KSLOC. In most cases, the quality of the data points was not very high. Several programs did not indicate the hours per man-month or phases of effort in the total effort expended. Further, the scope was non-standard (i.e., did not reflect SDR through FQT) and several programs used physical SLOC counts. Yet, there was a sufficient amount of high quality data available after normalization to be used for normalized regression and productivity factor analysis.

3.4.6 SILVER SASET VALIDATION DATABASE

This database came from an undated SASET validation study. There was no published report associated with this database; however, a presentation of the results of the study was given in 1990. The Silver Database, in its raw form, contained 42 CSCIs from 23 programs. The programs were developed from the 1970s to the mid-1980s. CSCI sizes ranged from 0.6 to 131 KSLOC, and program sizes ranged from 22 to 469 KSLOC. No information was provided about the programming languages used other than the top-level percentage of HOL and Assembly language.

Many of the data points were identical to those in the MITRE Non-Ada Database. To avoid duplication, NCCA screened each data point in the Silver Database to determine if it was already included in the MITRE Non-Ada Database. However, even if a duplicate was found in the Silver Database, additional information was often discovered. The Silver Database included additional fields such as program and/or CSCI name, CDR date, date of Initial Operational Capability (IOC). In one case, the MITRE Non-Ada Database had only program-level data while its match in the Silver Database decomposed the program by CSCI. The inclusion of the more granular CSCI-level data should enable the analyst to make stronger analogies.

For screening purposes, NCCA identified the following key fields to compare:

- 1) Total SLOC
- 2) Effort
- 3) CSCI name with CSCI initials

Appendix B contains the 11 data points from the Silver Database which were duplicated in the MITRE Non-Ada Database.

The Silver Database defined SLOC as DSLOC, but no formal definition was provided for SLOC. Therefore, except for the programs matched to the MITRE Non-Ada Database, a “?” appears in Field #22 (Count) of the NCCA Raw Database for those programs in the Silver Database.

SLOC were decomposed by code condition (i.e., new, modified, and rehosted) and language (i.e., HOL and Assembly). The specific name of the HOL was not provided; therefore, Field #10 (Lang1) in the NCCA Raw Database listed “HOL” as the programming language. This excluded this data from any later analyses that required the name of a specific HOL as an input.

Unless the program matched a data point from the MITRE Non-Ada Database, the hours per man-month were also unknown. In the NCCA Raw Database, this means that Field #25 (MM.eq.152) contained a question mark for those data points.

Since the type of counting convention, scope of effort, programming language and hours per man-month were all unknown, unless the same data points were found in the MITRE Non-Ada Database, this database is useful only to develop non-normalized, top-level productivity factors.

3.4.7 REVIC RECALIBRATION DATABASE

The REVIC Recalibration Database contained program and CSCI-level data that was collected to recalibrate the REVIC effort and schedule equations. The study [12] relied on more than five different sources of data including the SMC Database, Front Range Ada Working Group, Wright Patterson Air Force Base, IITRI, and others (such as NASA, Boehm, and the Jet Propulsion Lab).

A total of 114 data points were compiled; 41 of these came from the SMC Database. The REVIC Database provided some program names, while the SMC Database did not. NCCA cross-referenced the REVIC Database with the SMC Database by comparing size, language, application, effort, and schedule. This was similar to the methods used to compare the Silver Database with the MITRE Non-Ada Database. Appendix B contains the REVIC data points duplicated in the SMC Database. The overall quality of data from this database was mixed. The 41 data points originating from the SMC Database were of a higher quality than the remaining data points and can be utilized to generate normalized tools. However, the remaining data is useful only for a top-level non-normalized analysis.

3.4.8 IITRI DATABASE

The IITRI Database [13] consisted of eight data points which also appear in the REVIC Recalibration Database described above. However, the IITRI Database provided additional information that was added to the NCCA Raw Database. IITRI collected data primarily to determine how several software cost estimating models compare when estimating Ada development programs. The report also specifically tracked the amount of Ada experience each programmer possessed. This experience ranged from zero to five years. However, it did not track the quantity of programs that the programmer had developed in Ada.

Some of the data points were not 100 percent Ada (i.e., Assembly language was also used). CSCI sizes ranged from 18.3 to 480 KSLOC. Several types of software were represented, including C³, avionics, and Ada support tools. One data point reflected commercial development standards, while the rest were developed based on various DoD standards, including DoD-STD-2167A.

It appears that IITRI defined these data points as program-level, yet the REVIC study showed them as CSCIs. Schedule information did not appear in the data summary tables in the IITRI report; however, page 3-16 of the report provided actual schedules. IITRI made a point to state the scope of effort (SDR through FQT), but never mentioned whether the schedule was also specific to these review dates. NCCA classified these data points as CSCIs, following the REVIC Database's format. If these points turn out to be programs and the schedule's phasing is comparable to the effort phasing, an additional eight data points can be added to the normalized software schedule estimating database. (See Section 8 - **Schedule Analysis** for a discussion of the schedule estimating effort.)

The IITRI data should be used with caution for the following reasons:

- 1) It is not precisely known if these are programs, CSCIs, or a mixture of the two.

- 2) Data point #5 was commented physical SLOC. IITRI spoke to the software developer and decreased the SLOC count by 20 percent to convert from physical SLOC to logical SLOC. Reference [13] did not state whether the count shown was the adjusted number or the raw number. In any case, NCCA removed the 20 percent discount factor and classified this point as "CP" in Field #22 (Count).
- 3) Data points #7 and #8 showed SLOC without indicating if they were new. The other six points were labeled "New Ada" or "New Assembly." The lack of "New" in front of data points #7 and #8 may indicate these were raw totals. However, since these data points were treated as 100 percent new SLOC in the REVIC Database, NCCA treated them as 100 percent new SLOC.

Overall, half of this data was useful only to develop top-level non-normalized factors.

3.5 RESULTS

As a result of the extraction of data points from the aforementioned databases, the NCCA Raw Database contains a total of 457 records. Each record contains 73 attribute fields, although not all fields are completed for each record. These fields describe various attributes of the program, including size, effort, schedule, language, scope, process, and many others. The database contains many different DoD and NASA software development programs at both the program- and CSCI-level. Not all of the data is of the quality NCCA requires. However, there are enough well-defined data points for the analyst to perform credible software cost analyses. Table 3-2 presents top-level comparisons of the program- and CSCI-level data:

| Characteristics | Program-Level | CSCI-Level |
|---------------------------|---------------|---------------|
| Number of Data Points | 151 | 306 |
| Number of Ada Data Points | 42 | 134 |
| Size Range (KSLOC) | 2.3 - 1,800.0 | 0.4 - 595.1 |
| Effort Range (MM) | 9 - 26,500.0 | 1.7 - 6,593.0 |
| Amount of Reused (%) | 0 - 100 | 0 - 96 |

Table 3-2: NCCA Raw Database Summary

Table 3-3 provides a comparison of the source databases by program and CSCI. Note the smaller number of data points for the program-level data. The program-level data points were extracted primarily from NASA and Navy Internal sources, while the CSCI-level data was extracted primarily from Navy Internal, REVIC Recalibration and SMC source databases.

The programs represented by these data points were written in several different programming languages. With the advent of fewer government mandates, programming in languages other than Ada will increase. For example, commercial vendors currently rely heavily on the C programming languages. The NCCA Raw Database does not contain any programs which used a 4GL.¹⁸ Table 3-4 compares the primary programming languages by program and CSCI. Note that there were some data points where the programming language was unknown.

The NCCA Raw Database covers diverse mission areas. However, the MIS area is not well represented. This is the most challenging area to collect data, as it is more sensitive to rapid

¹⁸ Refer to the NCCA Issue Paper "Fourth Generation Languages" for a discussion of this topic.

Section 3 - Software Database

changes in the commercial sector. The problem is compounded by DoD's increasing emphasis on utilizing COTS software.¹⁹ Table 3-5 compares the data by mission area.

| Source Database | Number of Data Points | | |
|----------------------------|-----------------------|------|-------|
| | Program | CSCI | Total |
| 1. MITRE Non-Ada | 21 | 51 | 72 |
| 2. MITRE Ada | 17 | 13 | 30 |
| 3. SMC | 17 | 62 | 79 |
| 4. NASA SEL | 33 | 4 | 37 |
| 5. Navy Internal | 50 | 77 | 127 |
| 6. Silver SASET Validation | 12 | 35 | 47 |
| 7. REVIC Recalibration | 1 | 56 | 57 |
| 8. IITRI Report | 0 | 8 | 8 |
| Total | 151 | 306 | 457 |

Table 3-3: NCCA Raw Database Data Sources

| Primary Language | Program-Level | CSCI-Level |
|------------------|---------------|------------|
| Ada | 42 | 134 |
| Assembly | 19 | 36 |
| Atlas | 0 | 3 |
| C | 4 | 20 |
| CMS-2 | 12 | 21 |
| COBOL | 3 | 7 |
| FORTRAN | 48 | 20 |
| JOVIAL | 8 | 33 |
| Other | 12 | 31 |
| Unknown | 3 | 1 |

Table 3-4: Summary of the NCCA Raw Database by Programming Language

| Mission Area | Program-Level | CSCI-Level |
|--------------|---------------|------------|
| ASW | 4 | 55 |
| C3 | 47 | 114 |
| EW | 6 | 0 |
| MINE | 3 | 0 |
| MIS | 7 | 16 |
| MISSILE | 2 | 7 |
| RADAR | 14 | 32 |
| SIM | 42 | 12 |
| SONAR | 7 | 1 |
| TORP | 2 | 0 |
| UUV | 2 | 2 |
| UNKNOWN | 15 | 67 |

Table 3-5: Summary of the NCCA Raw Database by Mission Area

Four platform types are represented in the database. The majority of the systems are ground systems; however, there are a substantial number of programs and CSCIs where the platform type is unknown. Platform types are summarized in Table 3-6.

| Platform | Number of Data Points | |
|----------------|-----------------------|------|
| | Program | CSCI |
| Air | 21 | 17 |
| Ground | 78 | 118 |
| Ship | 29 | 69 |
| Unmanned Space | 0 | 11 |
| Unknown | 23 | 91 |

Table 3-6: Summary of the NCCA Raw Database by Platform

¹⁹Refer to the NCCA Issue Paper "Commercial-Off-The-Shelf Integration" for a discussion of this topic.

While the results seem impressive, users are cautioned that the tables are based on the non-normalized data set (all 457 points). Sections 4 and 5 - **Effort Analysis: Significant Drivers and Normalized Regressions**, thoroughly explain the process of filtering the data points to obtain a normalized, standardized set of data points. Further subdivision of the NCCA Raw Database may reduce the number of data points within a specific mission area to zero.

3.6 CONCLUSIONS

NCCA invested extensive effort to compile all readily available data into one centralized database. The NCCA Raw Database represents a strong foundation upon which to build further capability and robustness. Overall, the database can be used to produce top-level non-normalized productivity factors, normalized productivity factors or effort regressions, and schedule estimators. With the future addition of high quality data, NCCA plans to use the database to track productivity improvements over time, develop schedule estimates at the CSCI-level, investigate the effect of development processes and standards on productivity, and investigate the productivity of 4GLs.

Finally, the authors would like to point out the continued scarcity of historical software data both in the Navy and throughout the government. While the 457 data points available in the NCCA Raw Database sound impressive, the data reflects a 25-year period of data collection. In that time, thousands of programs have been developed with perhaps tens of thousands of subcomponents. DoD has not performed well in collecting software data.

There is no formal mechanism in place to collect this data in a standardized, well-defined manner; therefore, cost analysts should make every attempt to obtain high quality data and devise better metrics to measure software products and processes. There are many factors and reasons why a software program can go awry and fail. Poor schedules²⁰ and improper budgets should not be among them.

3.7 FUTURE EFFORTS

It is clear that more can be done to improve the historical database. While compiling the current version of the NCCA Raw Database, several potential improvements and enhancements were identified:

- 1) **Maintenance:** The current database covers only software development. Software maintenance, which represents 50 to 70 percent of software life cycle cost, is not covered. The SMC Database contains additional fields to collect software maintenance information, but, as of Version 2.1, only two records in the database contain maintenance information. NCCA is currently working with NSWC, Dahlgren to remedy this situation by funding the collection and analysis of Navy software maintenance data.²¹
- 2) **4GL:** The current database does not contain any programs written in a 4GL. The MIS area will be the first to see the increased adoption of 4GLs. Further research into the literature is

²⁰ Refer to the NCCA Issue Paper "The Impacts of Schedule Slippage/Compression on the Software Development Effort" for a discussion of this topic.

²¹ Refer to the Technomics study, titled "Software Life Cycle Cost Process Model," dated April 1995.

needed to determine what is the best and easiest metric to measure 4GL programs. SLOC may no longer be sufficient because much of the code is generated automatically and in a non-sequential manner.

- 3) **Effort by Phase:** Most of the data points contain total effort only. However, some of them provide effort by major phase (e.g., design, code, unit test, integration, and system test). Still, more effort should be devoted to collecting the effort by phase, primarily because it is hypothesized that some phases are more sensitive to the SLOC count than others. Detailed Design and Code and Unit Test are two such phases. Regressions based only on these two phases may result in better statistics. Other phases may be appropriately estimated by utilizing a level-of-effort approach. Additionally, not all estimates require the same scope of phasing. If the effort is decomposed by phase, then general factors can be developed to either add or delete effort from the total.
- 4) **Effort by WBS:** The only databases to provide effort information by WBS were the SMC and SEL Databases. The other databases treated the WBS as a normalizing factor and tried to ensure that the total effort information included common WBS tasks. See reference [5] for more details. Collecting effort by specific WBS greatly enhances the ability to standardize effort data.²² It also allows the analyst to generate standard factors by WBS. These are useful in addressing the impact of process and acquisition changes on cost, which are of prime interest in today's acquisition reform environment. For instance, if data requirement costs are reduced by 50 percent, how much savings will there be to the total software development cost?
- 5) **Process:** Most of the fields in the NCCA Raw Database describe the software product, but fail to describe the process that created the software. If a set of objective attributes can be found to define what process is used to generate the software product, then stronger analogies and tighter regressions may result. One attribute already captured is the development process (i.e., waterfall, incremental, spiral, etc.). This, however, is only the tip of the iceberg. Attributes like those set forth in the Capability Maturity Model²³ should be explored. Other software models attempt to capture these attributes, most, however, are too subjective.
- 6) **Function Points:** Another way to estimate the overall size of the software product is to count function points. Function point enthusiasts claim it is easier for a software engineer to estimate how much functionality his or her software product will have than to develop an estimate of the SLOC that need to be produced. In the past, it was claimed that weapon systems were not good candidates for function point counts. Capers Jones attempted to address this shortfall through the creation of feature points (which take into account the number of algorithms in a program). Today not only has the function point counting practice matured, but modern military systems are looking more and more like sophisticated MIS systems and less and less like black boxes. This may enable greater application of function points in estimating weapon systems.
- 7) **Tools:** Great advances have been made in software development tools over the past few years. What effect do modern tools like Visual Basic or PowerBuilder have on the overall

²² Refer to the NCCA Issue Paper "The Relationship Between MIL-STD-881B and DoD-STD-2167A" for a discussion of this topic.

²³ Refer to the NCCA Issue Paper "Software Engineering Institute Capability Maturity Model" for a discussion of this topic.

productivity of software developers? How can they be measured? These answers may unlock the door to many other productivity issues.

- 8) **Effort Associated with Deleted Code:** As discussed previously, many companies do not track the amount of deleted code. The effort to understand and remove faulty code may be more difficult than currently assumed. By tracking the effort associated with deleted code, regressions and productivity metrics may experience great improvements.
- 9) **Effort by Type:** Similarly, if companies captured not only the effort and size associated with the delivered operational code but also the effort associated with non-delivered code, improvements in cost estimating tools may result.

EFFORT ANALYSIS: SIGNIFICANT DRIVERS

4.1 INTRODUCTION

This section of the handbook documents the methodology and procedures used to determine which software attributes drive productivity. Identification of the significant productivity drivers aided in the development of data normalization procedures as well as providing guidelines for the development of effort estimating tools (Section 5 - **Normalized Regressions**).

The analysis was performed on four levels; each level tested a different set of attributes. NCCA chose to limit the analysis to objective metrics since 1) the analysis was geared toward the novice software cost estimator and 2) subjective metrics would increase the uncertainty.

This portion of the handbook is organized into the following sections:

- Raw Data
- Methodology and Results
- Conclusions
- Weaknesses
- Future Efforts

4.2 RAW DATA

All data used in this section came from the NCCA Raw Database (see Section 3 - **Software Database**). However, analysis of the NCCA Raw Database in its non-normalized state does not provide meaningful results. Many of the records in the database used different units of measure for items like size, effort, and schedule. To rectify this, NCCA decided to compile a normalized database. There are two possible approaches for constructing the normalized database.

The first approach is to keep all data points, but adjust them as necessary to obtain consistent units of measure. If the NCCA normalized database required each data point to include all phases of the software development effort and a data point did not include the effort for the Software Requirements Analysis Phase, then the data point would have to be adjusted to include the Software Requirements Analysis Phase. The question then becomes, "What factors should be used to make these adjustments?" The answer can vary tremendously and may be based more on engineering judgment than on empirical study. Using this first approach may introduce an additional layer of uncertainty into an already highly volatile database.

The second approach is to filter out any data points that do not meet the specified criteria (e.g., exclude all Assembly data points, include only ship programs, etc.). The main advantage of this approach is that no additional uncertainty is introduced into the analysis. The main

disadvantage to this approach is the reduction in the amount of data. In other words, the more specific the filter, the fewer data points that remain at the end of the normalization process.

To minimize uncertainty, NCCA chose to normalize the database by filtering out those data points which did not satisfy the specified criteria (i.e., no factors were used to make adjustments to the data points). The remainder of this section of the handbook focuses on determining significant productivity drivers. The results of the analysis will determine how the data must be filtered to constitute a “normalized” database.

4.3 METHODOLOGY AND RESULTS

NCCA conducted analysis on four levels. For each level of analysis, five-steps were followed to determine which attributes drive productivity. First, NCCA determined which attributes should be tested as possible productivity drivers. Second, the relevant data sets were filtered from the NCCA Raw Database. NCCA created eight data sets for the Level One analysis, three data sets for the Level Two analysis, 17 data sets for the Level Three analysis, and 17 data sets for the Level Four analysis. Third, productivity (Hours/ESLOC) was calculated for each data point. Fourth, average productivity, standard deviation, and coefficient of variation (CV) were calculated for each set of data. Fifth, statistical tests were conducted on the sets of data to determine whether the data sets were statistically different. See Appendix C for definitions of the statistical measures.

4.3.1 LEVEL ONE

Level One analyses were top-level and conducted to determine whether a) mission (MIS versus weapon system); b) counting convention (physical versus logical); c) language (Assembly versus HOL); and/or d) phasing (SDR through FQT versus other life cycle phases) were significant productivity drivers.

In this section, the methodology and the results for each attribute tested for the Level One analysis will be discussed. The results from Level One are in Appendix C. Throughout this section, tables (such as Table 4-1) demonstrate exactly how the data was filtered from the NCCA Raw Database. The italicized words are the field names from the NCCA Raw Database used for the filters. The non-italicized words are the criteria utilized to filter the database. Additionally, figures (such as Figure 4-1) show the resulting data sets after the filtering criterion was applied (the double boxes depict the “normalized” database filters).

4.3.1.1 MISSION

METHODOLOGY

The operational requirements of MIS and weapon systems are quite different. Weapon systems typically require real-time processing. In addition, failure of a weapon system could result in failure of the mission or in loss of life. Thus, the reliability requirements for weapon systems are very demanding. In contrast, MIS are not typically real-time, mission critical systems, and failure of the system does not result in loss of life. Thus, the reliability requirements for a MIS

Section 4 – Effort Analysis: Significant Drivers

are much less stringent. Due to these differences, it was expected that the productivity to develop weapon system software would be lower than that to develop MIS software.

To determine whether mission drives productivity, two data sets (weapon system and MIS) were filtered from the NCCA Raw Database, as Figure 4-1 illustrates. The 112 data points deleted did not meet the filtering criteria (i.e., mission was not known) or they were deemed outliers, as discussed below.

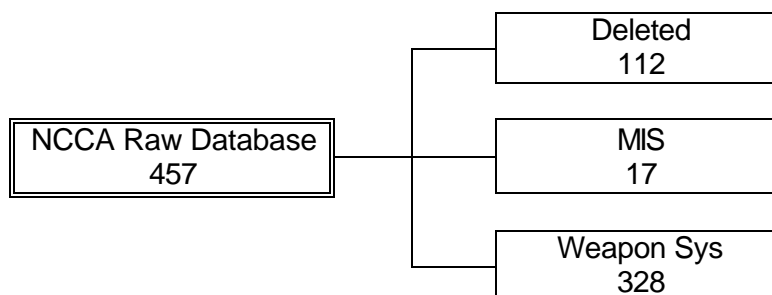


Figure 4-1: Mission Data Sets

The original MIS data set consisted of 23 MIS programs (or data points) from the NCCA Raw Database with known code condition.²⁴ The following six data points were then deleted from the data set resulting in a final MIS data set of 17 data points: NCCA-39, NCCA-43, NCCA-55, NCCA-74, NCCA-92, and NCCA-157. These points were deleted due to the possibility that they were duplicates and reflected effort that was allocated to the CSCI-level.

Table 4-1 shows the filters used to obtain the MIS data set from the NCCA Raw Database. The weapon system data set (328 data points) consists of all weapon system programs from the NCCA Raw Database with known code condition.

| MIS | Weapon System |
|---|---|
| <i>New</i> ≠ blank <i>Mission</i> = MIS (Eliminated 6 outliers) | <i>New</i> ≠ blank ²⁵ <i>Mission</i> ≠ MIS or blank ²⁶ |

Table 4-1: MIS and Weapon System Data Sets

The productivity expressed in Hours per equivalent new SLOC (ESLOC), was then calculated for each data point. The measure, ESLOC, is a means to normalize SLOC counts to reflect the fact that new code requires greater development effort than not new code. A popular method to determine ESLOC is discussed in reference [5]. The method is based on engineering judgment and an assumed distribution of effort between design, code, and test. The term Adaptation Adjustment Factor (AAF) describes the overall weight given to the adapted (i.e., not new) SLOC.

$$AAF = 0.4 DM + 0.3 CM + 0.3 IM$$

²⁴The code condition must be known to perform the ESLOC calculation.

²⁵This field must be filtered manually. There is one program with zero percent new code. LOTUS treats this data point as if there was a blank in the field and eliminates it.

²⁶All data points with a blank Mission Field were also eliminated since they could be MIS programs.

Section 4 – Effort Analysis: Significant Drivers

Where DM (% design modified), CM (% code modified), and IM (% of integration and test required) are percentages of the adapted software's code that needs to be redesigned, recoded and retested. Thus, if DM equals 100 percent, the adapted SLOC undergoes 100 percent redesign. If DM equals zero, the adapted SLOC undergoes no redesign. After determining AAF, ESLOC are calculated by the following equation:

$$\text{ESLOC} = \text{New SLOC} + (\text{AAF} * \text{Adapted SLOC})$$

There are three problems with this methodology:

- 1) DM, CM, and IM must be subjectively estimated by an engineer at the start of the project, presumably when little information is available.
- 2) With the advent of Best Commercial Practices, many standards and requirements are being relaxed (which might decrease the amount of documentation and testing required). This would impact the phasing distribution assumed in the AAF equation above.
- 3) There is no uncertainty range around any of the assumed values (i.e., around the coefficients or around DM, CM, and IM).

These problems make it difficult to assess the overall variance of the effort estimate and therefore, to budget to the true most likely estimate.

Due to the problems associated with the subjectivity of this methodology, NCCA adopted an alternative approach. The NCCA Raw Database contains a mixture of different kinds of development. They include:

- 1) 100 percent new program developments
- 2) Programs with various levels of reused code (both external and internal)

Based on the composition of the NCCA Raw Database, NCCA's alternative approach calculates ESLOC empirically. NCCA used an Efactor²⁷ (Equivalent Code Conversion Factor) to convert reused SLOC into ESLOC. The Efactor weights reused SLOC as a percentage of new SLOC. The term ESLOC is defined as follows:

$$\text{ESLOC} = \text{New SLOC} + (\text{Efactor} * \text{Reused SLOC})$$

Efactors are iteratively derived using a simple spreadsheet model. The model performs a "tradeoff analysis." During the analysis, the model assigns the Efactor a value between zero and one, and then solves for the X variable, ESLOC. After the productivity, expressed in ESLOC, is calculated, the CV is computed. (See Appendix C for further details.) The Efactor is then changed and the productivity and CV are re-calculated. This continues until the productivity and CV corresponding to each Efactor value, in increments of 0.01 between zero and one, have been computed. The results are then analyzed to determine the value of the Efactor that produced the productivity with the lowest CV.

²⁷Note: $0 \leq \text{Efactor} \leq 1$

Section 4 – Effort Analysis: Significant Drivers

Table 4-2 summarizes the strengths and weaknesses of the two different approaches to estimating ESLOC.

| ESLOC Method | Strengths | Weaknesses |
|--|--|--|
| Engineering Judgement-Typical Approach | <ol style="list-style-type: none"> 1) Based on a technical assessment of effort to be performed 2) Estimate would be company specific | <ol style="list-style-type: none"> 1) Subjective assessment 2) Information required to make assessment may not be available early in the program's development 3) Assumed distribution of effort 4) No uncertainty around estimate (%redesign, %recode, and %retest are an assumed distribution) |
| Empirical - NCCA Approach | <ol style="list-style-type: none"> 1) Objective approach 2) Approach can be applied early in development cycle 3) Reflects industry averages (if industry average vice contractor-specific data is used) 4) Uncertainty around estimate captures Efactor uncertainty | <ol style="list-style-type: none"> 1) No specific uncertainty around Efactor 2) Not contractor-specific unless underlying data is 3) Dependent upon mapping of reused SLOC into correct fields (Two Efactor equation only) |

Table 4-2: ESLOC Methods (Strengths and Weaknesses)

Productivity was calculated for each data point as follows:

$$\text{Productivity} = \frac{\text{MM} * (\text{Hours/MM})}{\text{ESLOC}}$$

where: MM is equal to the number of man-months of effort expended to develop the software program; Hours/MM is equal to the number of hours in a man-month; and ESLOC is the number of equivalent new SLOC.

Most of the source databases tracked effort in man-months. For this level, a 152-hour man-month was assumed for those data points that did not provide the hours per man-month rate.

The average productivity, standard deviation, and CV were then calculated for each data set. In addition, NCCA performed statistical tests on the data to determine which metrics drive productivity. The t-test and the Mann-Whitney U test were used to determine whether the productivities of the two data sets were statistically different. For both tests, NCCA assumed a two-tailed test and a confidence level of 95 percent ($\alpha = 0.05$).

The t-test was used to test the hypothesis that the two data sets are from populations with the same mean. The test assumes two independent, normal populations with unknown means and unknown but equal variances ($s_1^2 = s_2^2 = s_3^2$). See Appendix C for details on the t-test.

The Mann-Whitney U test is a nonparametric alternative to the t-test appropriate when sample sizes are small.²⁸ It is a ranking test which assumes that if the two data sets are actually drawn from the same population, then the observations will be dispersed throughout (i.e., one data set is not concentrated among the smaller values, while the other is concentrated among the larger values). See Appendix C, and reference [14] for details on the Mann-Whitney U Test. The results from the Mann-Whitney U Test are also contained in Appendix C. This methodology was duplicated for all remaining levels of analyses which follow.

²⁸NCCA utilized the Mann-Whitney U test when any data set consisted of 20 or fewer data points.

RESULTS

The software development productivity for MIS programs (0.824 Hours/ESLOC) was statistically higher than that for weapon systems (1.781 Hours/ESLOC). Therefore, since the mission (or domain) of the system is a significant productivity driver, MIS programs should not be combined with weapon system programs. Table 4-3 shows the detailed results and corresponding statistics.

| Metric | Efactor | # of Data Points | Average Productivity (Hours/ESLOC) | CV | Test | Equal? |
|----------------|---------|------------------|------------------------------------|------|--------------|--------|
| MIS | 0 | 17 | 0.824 | 72% | Mann-Whitney | No |
| Weapon Systems | 0.46 | 328 | 1.781 | 128% | | |

Table 4-3: Level One Statistical Results (Mission)

4.3.1.2 COUNTING CONVENTION

METHODOLOGY

There are many ways to count SLOC in a program, and each produces a different result as discussed in Section 3 - **Software Database**. Each carriage return is counted as a line when counting physical SLOC. When counting logical SLOC, each complete command is counted as a line, regardless of how many physical lines the command takes. Logical SLOC are typically more reflective of the true effort associated with a function, because the count is not influenced heavily by coding style. Due to these differences, it was expected that software programs sized by a physical code count would appear to be more productive than programs sized by a logical code count.

To determine whether counting convention drives productivity, two data sets (logical and physical) were developed. Since mission was proven to be a productivity driver and this handbook is concerned with weapon systems, MIS programs were excluded from the remaining data sets. The two data sets developed for this test were filtered from the weapon system data set, as demonstrated in Figure 4-2.

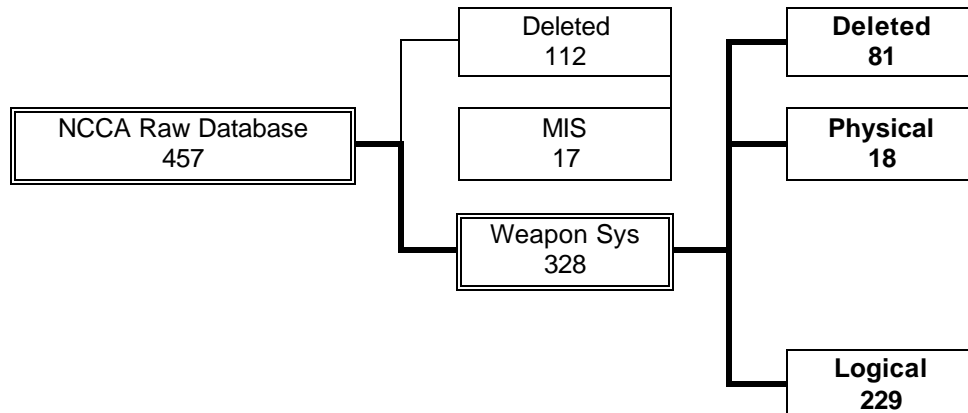


Figure 4-2: Counting Convention Data Set

The physical data set consisted of weapon system programs with known code condition that were sized by a physical code counting convention. The logical data set consisted of weapon

Section 4 – Effort Analysis: Significant Drivers

system programs with known code condition that were sized by a logical code counting convention. See Table 4-4 below.

| Physical | Logical |
|-------------------------------|----------------------------------|
| <i>Count = P</i> | <i>Count = L</i> |
| <i>Mission ≠ MIS or blank</i> | <i>Mission ≠ MIS or blank</i> |
| <i>New ≠ blank</i> | <i>New ≠ blank</i> ²⁹ |

Table 4-4: Physical and Logical Data Sets

RESULTS

The software development productivity for programs sized using a physical code count (0.735 Hours/ESLOC) was statistically higher than that for programs sized using a logical code count (1.739 Hours/ESLOC). Therefore, counting convention is a productivity driver and should be considered when estimating productivity. Table 4-5 shows the detailed results and corresponding statistics.

| Metric | Efactor | # of Data Points | Average Productivity (Hours/ESLOC) | CV | Test | Equal? |
|----------|---------|------------------|------------------------------------|------|--------------|--------|
| Physical | 0 | 18 | 0.735 | 104% | Mann-Whitney | No |
| Logical | 0.53 | 229 | 1.739 | 124% | | |

Table 4-5: Level One Statistical Results (Counting Convention)

4.3.1.3 LANGUAGE

METHODOLOGY

Assembly is a second-generation language (2GL) and is one step above machine language. HOLs are third-generation languages (3GLs) and are closer to spoken language than 2GLs. HOLs were developed to make writing and understanding programs easier. Additionally, programs that used significant amounts of Assembly probably did so because of severe constraints on memory or timing requirements. This extra level of complexity is in contrast to other programs that did not have these constraints. Based on this reasoning, it was expected that the productivity to develop code written in an HOL would be higher than that to develop code written in Assembly.

To determine whether language level drives productivity, two data sets, which also reflect the results of the mission and counting convention productivity analyses previously discussed, were developed from the NCCA Raw Database. Thus, the two data sets developed for this test were filtered from the logical data set, as demonstrated in Figure 4-3.

The HOL data set consisted of weapon system programs with known code condition that were sized by a logical code count and written primarily in an HOL (greater than or equal to 70 percent). NCCA used HOL greater than 70 percent as the cutoff, based on recent programs that NCCA reviewed. Sensitivity analyses were performed at 80 percent HOL and 90 percent HOL. On the whole, the 80 percent filter produced comparable results to the 70 percent criterion, but with fewer data points. The 90 percent HOL filter was so restrictive that, at the program-level, entire database sources were deleted. Hence, to retain as many data points as possible, while still addressing the impacts of Assembly, NCCA chose 70 percent as the cutoff

²⁹This field must be filtered manually. There is one program with zero percent new code. LOTUS treats this data point as if there was a blank in the field and eliminates it.

Section 4 – Effort Analysis: Significant Drivers

percentage. This percentage actually makes the resulting differences smaller (i.e., if 100 percent HOL programs had been used in the data set rather than greater than 70 percent HOL programs, the resulting differences would have been larger).

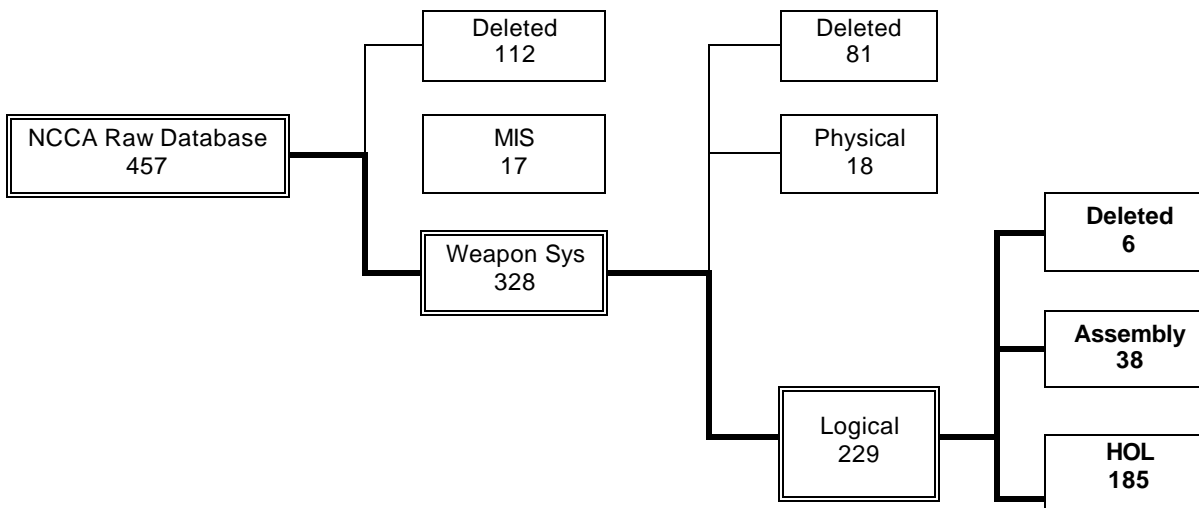


Figure 4-3: Language Data Set

The Assembly data set consisted of weapon system programs with known code condition that were written entirely in Assembly (zero percent HOL). See Table 4-6 below. Counting convention was not filtered for the Assembly data set because programs written in Assembly are always sized by a logical code counting convention.

| HOL | Assembly |
|--------------------------------|-----------------------------|
| $HOL \geq 0.7$ | $HOL = 0$ |
| Count = L | Mission \neq MIS or blank |
| Mission \neq MIS (or blank) | New \neq blank |
| New \neq blank ³⁰ | |

Table 4-6: HOL and Assembly Data Sets

RESULTS

The software development productivity for programs written in Assembly (3.990 Hours/ESLOC) was statistically lower than that for programs written in an HOL (1.860 Hours/ESLOC). Therefore, since language is a significant productivity driver, programs written in Assembly must be treated separately from programs written in an HOL. See Table 4-7 for the detailed results and corresponding statistics.

| Metric | Efactor | # of Data Points | Average Productivity (Hours/ESLOC) | CV | Test | Equal? |
|----------|---------|------------------|------------------------------------|-----|--------|--------|
| Assembly | 0.69 | 38 | 3.990 | 98% | t-test | No |
| HOL | 0.04 | 185 | 1.860 | 83% | | |

Table 4-7: Level One Statistical Results (Language)

³⁰This field must be filtered manually. There is one program with zero percent new code. LOTUS treats this data point as if there was a blank in the field and eliminates it.

4.3.1.4 PHASING

METHODOLOGY

Total effort is the accumulation of work performed over a specific period of time. Depending on how the start and stop points are defined, a different accumulation of effort results. DoD 2167A defines 11 phases of system development, including software development (see shaded area in Figure 4-4). Assuming there is some software-related effort in each phase, a different amount of effort will result when particular phases are included or excluded.

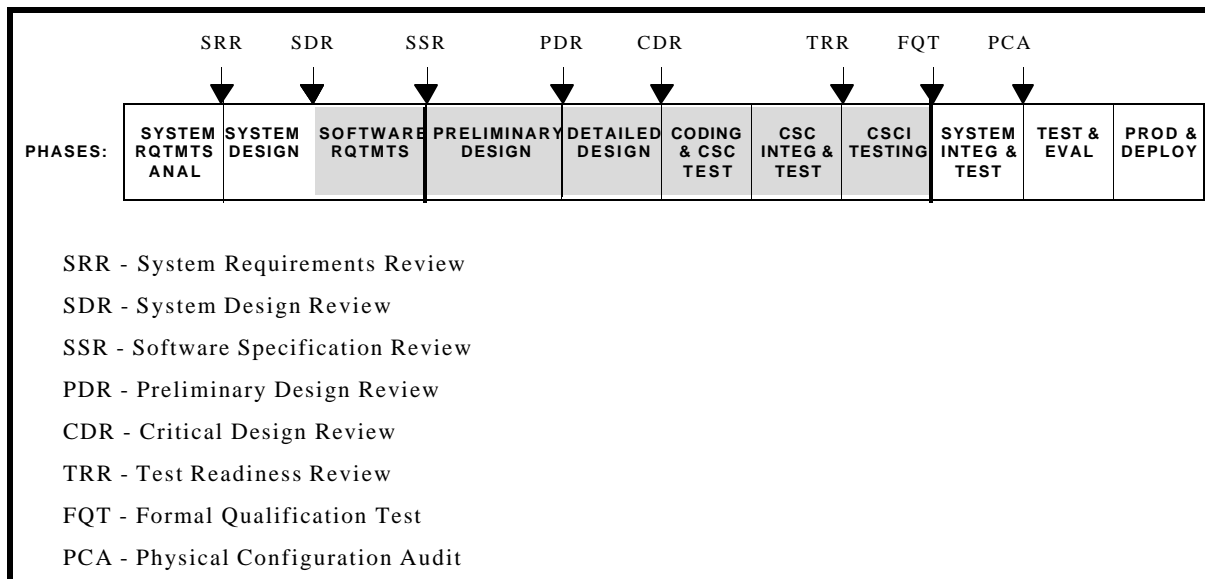


Figure 4-4: Phases of Software Development

For example, consider software requirements. Sometimes software requirements are captured in the System Requirements Analysis Phase rather than in the Software Requirements Analysis Phase. Sometimes software requirements are performed by the government rather than by a contractor. NCCA conducted a test to determine whether the software requirements effort was a significant portion of the total life cycle development.

To determine whether phasing drives productivity, two data sets were developed from the NCCA Raw Database which reflect the results from the mission, counting convention and language productivity analyses previously discussed. The two data sets developed for this test were filtered from the HOL data set, as demonstrated in Figure 4-5.

The normalized data set consisted of weapon system programs with known code condition that were sized by a logical code count, written primarily in an HOL, normalized to a 152-hour man-month, and included the effort associated with the phases of SDR through FQT (see the double boxes in Figure 4-5). Once the explicit phases are specified, the hours per man-month rate associated with the phases must be known. Therefore at this level, and henceforth NCCA excluded data points where the hours per man-month were unknown. All other data points had effort normalized to 152 hours per man-month.

Section 4 – Effort Analysis: Significant Drivers

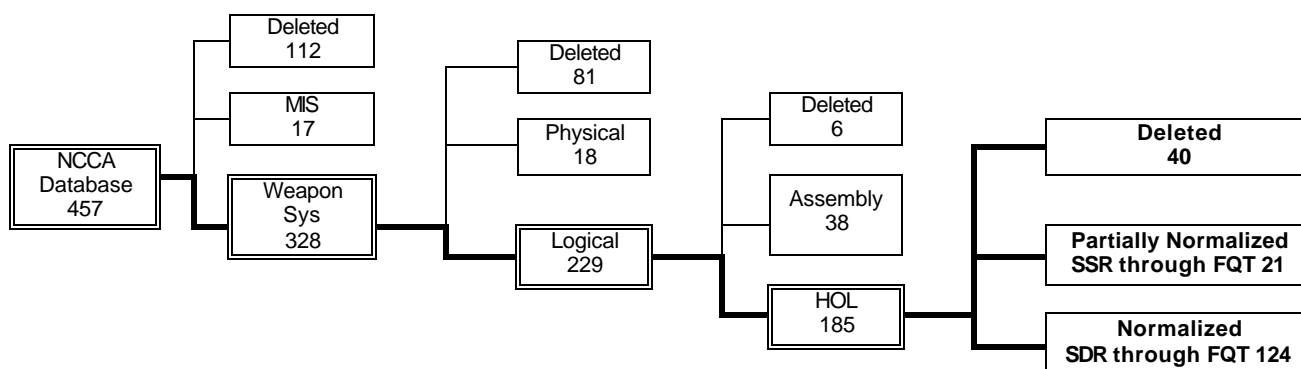


Figure 4-5: Phasing Data Set

The partially normalized data set consisted of weapon system programs with known code condition that were sized by a logical code count, written primarily in an HOL, normalized to a 152-hour man-month, and included the phases of SSR through FQT. See Table 4-8.

| Normalized | Partially Normalized |
|----------------------------------|-------------------------------|
| <i>Mission</i> ≠ MIS or blank | <i>Mission</i> ≠ MIS or blank |
| <i>Count</i> = L | <i>Count</i> = L |
| <i>HOL</i> ≥ 0.7 | <i>HOL</i> ≥ 0.7 |
| <i>New</i> ≠ blank ³¹ | <i>New</i> ≠ blank |
| <i>MM.eq.152</i> = Y | <i>MM.eq.152</i> = Y |
| <i>REQ</i> = 1 | <i>REQ</i> = 0 |
| <i>PD</i> = 1 | <i>PD</i> = 1 |
| <i>DD</i> = 1 | <i>DD</i> = 1 |
| <i>CUT</i> = 1 | <i>CUT</i> = 1 |
| <i>CSC TST</i> = 1 | <i>CSC TST</i> = 1 |
| <i>CSCI TST</i> = 1 | <i>CSCI TST</i> = 1 |
| <i>SIT</i> = 0 | <i>SIT</i> = 0 |
| <i>OTE</i> = 0 | <i>OTE</i> = 0 |

Table 4-8: Normalized and Partially Normalized Data Sets

RESULTS

The software development productivity for partially normalized (SSR through FQT) programs (1.096 Hours/ESLOC) was statistically higher than that for normalized (SDR through FQT) programs (2.025 Hours/ESLOC). This test proves that the Requirements Analysis Phase (from SDR through SSR) is a significant portion of the software development life cycle; therefore, phase-specific data sets should be utilized when estimating productivity. Table 4-9 shows the detailed results and corresponding statistics.

| Metric | Efactor | # of Data Points | Average Productivity (Hours/ESLOC) | CV | Test | Equal? |
|----------------------|---------|------------------|------------------------------------|-----|--------|--------|
| Partially Normalized | 0 | 21 | 1.096 | 45% | t-test | No |
| Normalized | 0.03 | 124 | 2.025 | 86% | | |

Table 4-9: Level One Statistical Results (Phasing)

Since mission, counting convention, language, and phasing were all proven to be significant productivity drivers, NCCA used data from the “normalized” data set (124 data points) for the next three levels of tests. This means that all data utilized for the remainder of the analyses

³¹This field must be filtered manually. There is one program with zero percent new code. LOTUS treats this data point as if there was a blank in the field and eliminates it.

Section 4 – Effort Analysis: Significant Drivers

were weapon system programs with known code condition that were sized by a logical code count, written primarily in an HOL (greater than or equal to 70 percent), normalized to a 152-hour man-month, and included the phases of SDR through FQT. This was an attempt to eliminate as many known productivity drivers as possible so that any differences in productivity could be isolated to the remaining attributes being tested.

4.3.2 LEVEL TWO

The Level Two Analysis was conducted to determine whether productivity rates are significantly different for CSCI-level versus program-level versus one-CSCI program-level programs. One-CSCI programs are programs with only one CSCI.

METHODOLOGY

By definition SDR through FQT does not include System Requirements and System Integration. These tests were conducted to verify that fact in regards to the specific data points included in this analysis.

Three data sets were developed to conduct the Level Two analysis. The three data sets were filtered from the normalized data set (124 data points). The program data set consisted of 27 normalized program-level data points. The CSCI data set consisted of 93 normalized CSCI-level data points. The one-CSCI data set consisted of 4 normalized, one-CSCI program-level data points. Table 4-10 illustrates how the data was filtered from the NCCA Raw Database.³²

| Program | CSCI | 1CSCI |
|-----------------------|------------------|-----------------------|
| <i>CSCI? = N</i> | <i>CSCI? = Y</i> | <i>CSCI? = N</i> |
| <i>CSCI Count ≠ 1</i> | | <i>CSCI Count = 1</i> |

Table 4-10: Program, CSCI, and 1CSCI Data Sets

RESULTS

The software development productivity for CSCI-level development efforts was statistically equal to that for program-level development efforts. The software development productivity for one-CSCI program-level development efforts was statistically equal to that for both CSCI-level and program-level development efforts. However, the one-CSCI data set consisted of only four data points, which is probably too small a data set to be conclusive. Table 4-11 provides a summary of all Level Two results and corresponding statistics. These results can also be found in Appendix C.

| Metric | Efactor | # of Data Points | Average Productivity (Hours/ESLOC) | CV | Test | Equal? |
|---------|---------|------------------|------------------------------------|-----|--------------|--------|
| Program | 0.07 | 27 | 1.726 | 68% | t-test | Yes |
| CSCI | 0 | 93 | 2.102 | 88% | | |
| 1CSCI | 0.28 | 4 | 1.762 | 3% | Mann-Whitney | Yes |
| Program | 0.07 | 27 | 1.726 | 68% | | |
| 1CSCI | 0.28 | 4 | 1.762 | 3% | Mann-Whitney | Yes |
| CSCI | 0 | 93 | 2.102 | 88% | | |

Table 4-11: Level Two Statistical Results

³² In addition to the attributes listed in Table 4-6, the attributes listed in Table 4-8 (Normalized Data set) were also used as filters to create the data sets used for this analysis.

4.3.3 LEVEL THREE (PROGRAM) AND LEVEL FOUR (CSCI)

The Level Three analysis tested lower-level attributes for the program-level and the Level Four analysis tested lower-level attributes for the CSCI-level. The results from both analyses are summarized in Appendix C. Based on the Level Two results, Level Three and Four data sets could have been combined into one data set. However, they were kept separate in order to support the regression analysis, discussed in Section 5 - **Effort Analysis: Normalized Regressions**.

NCCA performed tests on the Level Three and Level Four data sets to determine whether code condition, platform, mission area, software class, status, mode, language, or size drives productivity for the program- and CSCI-levels, respectively.

METHODOLOGY

The following discusses the rationale behind testing each attribute:

- 1) **Code Condition:** A completely new program or CSCI should be inherently more difficult to develop than its complement because there is no completed product, not even designs or algorithms, to reuse. On the other hand, a program or CSCI that is not hundred percent new should be inherently simpler to develop because there exists some software product (i.e. document or source code) to reuse. Therefore, developing a line of reused code (modified, rehosted, or verbatim) should be more productive than developing a line of new code. One hundred percent new data points versus not hundred percent new data points were tested to verify this.
- 2) **Platform:** Air versus non-air data points and ground versus ship data points were tested to determine whether platform type drives productivity. Because air systems have more physical constraints than non-air systems, the productivity to develop software for air systems may be significantly lower. Ship versus ground systems were tested to ensure that any differences found between air and non-air systems were not due to differences in non-air systems (i.e., differences between ship and ground systems). Additionally, because ship systems have more physical constraints than ground systems, the software development productivity for ship systems may be significantly lower.
- 3) **Mission Area:** C³ software is the component of weapon system software that communicates, assimilates, coordinates, analyzes, interprets information, and provides decision support for military commanders. It provides instantaneous situation assessment, allowing for advantageous, timely positioning and decision making [15]. Because C³ systems are more software dependent, than non-C³ systems, their software development productivity may be lower.
- 4) **Software Class:** System software is designed for a specific software system, or family of software systems, to facilitate its development, operation, and maintenance [15]. Application software is specifically developed for the functional use of a computer system. Examples are battle management, weapons control, and database management software [15]. Due to these differences, the development productivity for system software may be significantly lower than for application software.

- 5) **Software Status:** Operational software is embedded in the system and is critical for mission accomplishment, while non-operational software is typically simulation or support code used to generate and test the operational software. Operational code was defined by reference [16] as the code delivered to the customer for end use. Non-operational software, which is typically not delivered, does not have to undergo the same level of rigor or documentation as operational software. This was also the case in some programs that reference [5] surveyed. Traditionally, code counting conventions like DSI included only delivered SLOC. If non-delivered SLOC (like test drivers) are written with the same level of care as the delivered software, references [5], [16], and [17] recommend they be counted as well.

For some programs NCCA reviewed within the source database, there was almost as much non-operational code as operational. None of the source databases explicitly tracked the effort associated with operational versus non-operational code. If the programs contained both types of code, but only the operational code was counted, then the effort associated with developing the non-operational code would be assigned to the operational code, thus increasing the overall cost per line of code for the entire program. Also, if the historical programs contained a mix of operational and non-operational code that was significantly different from today's mix, then inaccurate results will be obtained. Software status was tested to determine if these differences do in fact affect productivity.

- 6) **Software Mode:** This attribute is based on the original COCOMO model definition (see reference [5]) and attempts to account for the difficulties encountered when the software is required to adhere to strict system requirements. Embedded mode software is characterized by tight constraints and is usually forced to comply with the specification of the system. Therefore, changing the requirements of the system in order to solve software problems is difficult. With the advent of cheaper memory and faster processors, there is room in today's software to relax the constraints on software. However, there are some programs that will continue to push the performance envelope. Based on this rationale, the development productivity for embedded mode software may be lower than for semi-detached or organic mode software.
- 7) **Language:** In the past, DoD had mandated the use of Ada to standardize software development. DoD anticipated that standardization would result in significant savings in personnel, training, software reuse, and tools [15]. Ada and non-Ada data sets were tested to determine whether these anticipated savings have actually been realized.
- 8) **Size:** Small versus large programs were tested to determine whether software size drives productivity. In a large program, it is possible that the programmers become more productive as they learn more about the program and the programming language. This may make larger programs more productive. On the other hand, a small program may be simpler or easier to manage and, therefore, more productive. Due to the uncertainty of how productivity is impacted by program size, the small and large program data sets were tested for significance.

To determine whether code condition, platform, mission area, software class, status, mode, language or size are significant productivity drivers, 17 data sets were developed for both the Level Three and Level Four analyses. Since one-CSCI programs were statistically equal to both CSCIs and programs, the four one-CSCI data points were added to both the program and

Section 4 – Effort Analysis: Significant Drivers

the CSCI data sets for analysis. The 17 data sets used in the Level Three analysis were filtered from the program and one-CSCI data sets (31 data points), and the 17 data sets used in the Level Four analysis were filtered from the one-CSCI and one-CSCI data sets (97 data points), as demonstrated in Figure 4-6:

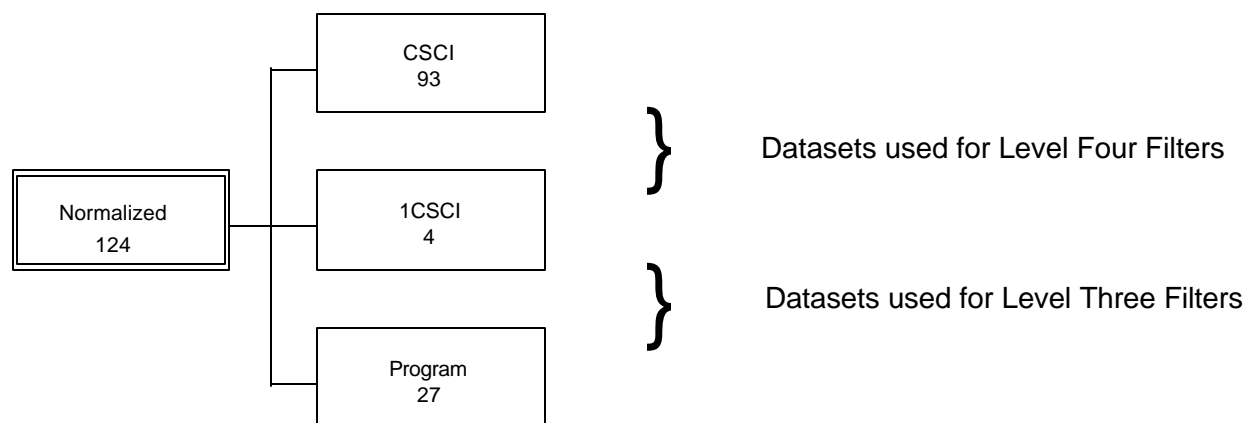


Figure 4-6: Level Three and Level Four Data Sets

Table 4-12 identifies the individual data sets, the corresponding number of data points within each data set, and the data filters utilized.

| Attribute | Data Set | Level Three (Program) | Level Four (CSCI) | NCCA Raw Database Filter |
|-----------------|--------------------------------------|------------------------|------------------------|---|
| Code Condition | 100% New <100% New | 8 23 | 32 65 | <i>New</i> = 1 <i>New</i> ≠ 1 |
| Platform | Ship Ground Air | 2 22 1 | 52 25 3 | <i>Platform</i> = Ship <i>Platform</i> = Ground <i>Platform</i> = Air |
| Mission Area | C ³ Non-C ³ | 11 20 | 32 65 | <i>Mission</i> = C3 <i>Mission</i> ≠ C3 |
| Software Class | System Application | 1 5 | 18 57 | <i>SWClass</i> = Sys <i>SWClass</i> = App |
| Software Status | Operational Non-Operational | 17 6 | 86 8 | <i>Status</i> = Op <i>Status</i> ≠ Op ³³ |
| Software Mode | Embedded Non-Embedded | 11 20 | 80 17 | <i>Mode</i> = Em <i>Mode</i> ≠ Em |
| Language | Ada Non-Ada | 7 24 | 49 48 | <i>Lang1</i> = Ada <i>Lang1</i> ≠ Ada |
| Size | Small Large | 15 ³⁴ 16 | 49 ³⁵ 48 | |

Table 4-12: Level Three and Level Four Data Sets

³³ Four additional data points must be eliminated manually because LOTUS will not filter them. The four data points are NCCA -417, NCCA -418, NCCA -426, and NCCA -454. These must be eliminated from the data set because they contain a mixture of operational and non-operational software.

³⁴ The program + 1CSCI data set was split in half according to program size (measured in ESLOC). An Efactor of 0.07 minimized the CV and was applied to each data point in the program data set, converting SLOC to ESLOC. The 15 smallest programs were designated the small data set, and the 16 largest programs were designated the large data set. The split was approximately 28,000 ESLOC.

³⁵ The CSCI + 1CSCI data set was split in half according to program size (measured in ESLOC). An Efactor of zero minimized the CV and was applied to each data point in the data set, converting SLOC to ESLOC. The 49 smallest programs were designated the small data set, and the 48 largest programs were designated the large data set. The split was approximately 16,000 ESLOC.

Section 4 – Effort Analysis: Significant Drivers

Data points were eliminated from the program or the CSCI data sets if there was a blank in the specific field being filtered. For example, the resulting data sets for the platform sort (program-level) consisted of only 25 of the 31 data points (2 ship, 22 ground, and 1 air). The six missing data points were eliminated because there were blanks in each of the six programs' Platform Field (i.e., for six programs, it was unknown whether the software was developed for a ship, ground or air program).

RESULTS

1) Code Condition

Level Three (Program): The software development productivity for a program with 100 percent new code was statistically lower than that for a program with less than 100 percent new code. The underlying data sets were fairly similar, although the "less than 100% New" data set was composed of more simulation and semi-detached software, while the "100% New" data set was composed of more C³ and embedded mode software. These differences in the underlying data sets were not as pronounced as in some of the other data sets tested (i.e., the "less than 100% New" data set had some C³ software, and the "100% New" data set had some semi-detached software). The "less than 100% New" data set was almost three times more productive than the "100% New" data set. Based on these results, 100 percent new programs appear to be more complex at the program-level.

Level Four (CSCI): The software development productivity for CSCIs with 100 percent new code was statistically lower than that for CSCIs with less than 100 percent new code. In contrast to the Level Three analysis, there were no major underlying data set differences that could have been driving this result. Since the CSCI-level result was consistent with the Level Three (program-level) result, it was concluded that code condition is a productivity driver. Table 4-13 provides a summary of results and corresponding statistics for both Level Three and Level Four code condition tests:

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|---------|-----------|---------|-------------|-----|--------------|--------|
| Program | 100% New | None | 2.485 | 43% | Mann-Whitney | No |
| | <100% New | 0.44 | 0.856 | 61% | | |
| CSCI | 100% New | None | 2.807 | 96% | t-test | No |
| | <100% New | 0.02 | 1.665 | 59% | | |

Table 4-13: Level Three and Level Four Statistical Results (Code Condition)

2) Platform

Level Three (Program): There were insufficient ship (two) and air (one) data points to test; therefore, a determination could not be made as to whether platform is a productivity driver at the program-level.

Level Four (CSCI): There were insufficient air (three) data points to test, so a determination could not be made as to whether air versus non-air platform is a significant productivity driver at the CSCI-level.

The software development productivity for ground CSCIs was statistically lower than that for ship CSCIs. However, the result may be due to differences in the underlying data sets. The ship data set was mostly Ada, anti-submarine warfare, application, embedded mode software,

Section 4 – Effort Analysis: Significant Drivers

while the ground data set was fairly well distributed. Additionally, the ship data set was composed primarily of verbatim code (51 percent on average), while the ground data set was composed primarily of new code (92 percent on average).

To make the code condition of the two data sets similar, the test was conducted again with all 100 percent new data points deleted. This time, the productivities of the two data sets were statistically equal and, in contrast to the previous results, ground CSCIs were more productive than ship CSCIs.

Although NCCA attempted to normalize the data sets by sorting out 100 percent new CSCIs, the two resulting underlying data sets were still quite different. The actual difference in productivity between the ship and ground data sets could not be isolated due to other differences in the underlying data sets. Therefore, a determination of whether ground versus ship platform is a significant productivity driver for the CSCI-level could not be made.

Table 4-14 provides a summary of results and corresponding statistics for both Level Three and Level Four platform tests:

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|-------|--------------------|---------|-------------|-----|--------------|--------|
| CSCI | Ground | 0 | 2.655 | 83% | t-test | No |
| | Ship | 0.02 | 1.522 | 53% | | |
| CSCI | Ground (<100% New) | 0.02 | 1.208 | 60% | Mann-Whitney | Yes |
| | Ship (<100% New) | 0.02 | 1.536 | 53% | | |

Table 4-14: Level Three and Level Four Statistical Results (Platform)

3) Mission Area

Level Three (Program): The software development productivity for C³ software was statistically lower than that for non-C³ software for the program-level. However, the result may have been due to other differences in the underlying data sets. The C³ data set consisted of more embedded mode programs, which could have decreased the average productivity. The non-C³ data set consisted almost entirely of support software programs, which could have increased the average productivity.

Additionally, the code condition of the two data sets was quite different. The C³ data set was 82 percent new (and zero percent verbatim) on average, while the non-C³ data set was only 53 percent new (and 21 percent verbatim) on average. This difference could have increased the productivity of the non-C³ data set, while decreasing the productivity of the C³ data set. It is impossible to tell whether the results were truly due to mission area differences (C³ versus non-C³), or were driven by other data set differences. Therefore, a determination could not be made as to whether mission area is a productivity driver for the program-level.

Level Four (CSCI): The software development productivity for C³ software was statistically lower than that for non-C³ software for the CSCI-level. However, similar to the program-level analysis, this result may have been due to other differences in the underlying data sets. The C³ data set was composed primarily of new code (on average 88 percent new) and the non-C³ data set was more evenly distributed (on average 51 percent new code and 49 percent reused code).

To eliminate the differences in code condition, the test was conducted again with the 100 percent new CSCIs. In contrast to the previous results, the productivity to develop C³ software was statistically equal to that for non-C³ software. However, there continues to be uncertainty

Section 4 – Effort Analysis: Significant Drivers

with this result due to additional differences in the underlying data sets. The 100 percent new C³ data set was composed almost entirely of embedded mode software, while the 100 percent new non-C³ data set was primarily composed of non-embedded mode software. This could possibly have decreased the average productivity of the C³ data set, while increasing that of the non-C³ data set.

Due to the underlying data set differences, it was impossible to determine whether mission area is a productivity driver for the CSCI-level. Table 4-15 provides a summary of results and corresponding statistics for both Level Three and Level Four mission area tests.

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|---------|-------------------------------|---------|-------------|------|--------------|--------|
| Program | C ³ | 0.15 | 2.411 | 44% | Mann-Whitney | No |
| | Non-C ³ | 0.18 | 1.003 | 52% | | |
| CSCI | C ³ | 0.02 | 2.740 | 74% | t-test | No |
| | Non-C ³ | 0 | 1.767 | 90% | | |
| CSCI | C ³ (100% New) | None | 2.785 | 79% | Mann-Whitney | Yes |
| | Non-C ³ (100% New) | None | 2.874 | 140% | | |

Table 4-15: Level Three and Level Four Statistical Results (Mission Area)

4) Software Class

Level Three (Program): There were only one system and five application data points to test; therefore, a determination could not be made as to whether software class is a significant productivity driver for the program-level.

Level Four (CSCI): The software development productivity for system software was statistically lower than that for application software for the CSCI-level. However, there is uncertainty associated with this result due to differences in the code condition of the underlying data sets. The system data set was composed primarily of new code (on average, 83 percent new), while the application data set was more evenly distributed (54 percent new code and 46 percent reused code). This may have decreased the productivity of system CSCIs, while increasing that of application CSCIs. In addition, the application data set was composed primarily of Ada programs that may also have skewed the results.

To decrease the differences in code condition between the two data sets, the test was conducted again with only 100 percent new CSCIs. Again, the software development productivity for system software was statistically lower than that for application software. The underlying data sets were similar, although both data sets were almost entirely composed of programs from the MITRE Non-Ada Database. Based on these test results, software class is a productivity driver for the CSCI-level. Table 4-16 provides a summary of results and corresponding statistics for both Level Three and Level Four software class tests.

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|-------|------------------------|---------|-------------|-----|--------------|--------|
| CSCI | System | 0 | 3.587 | 88% | Mann-Whitney | No |
| | Application | 0 | 1.850 | 64% | | |
| CSCI | System (100% New) | None | 4.241 | 91% | Mann-Whitney | No |
| | Application (100% New) | None | 2.684 | 53% | | |

Table 4-16: Level Three and Level Four Statistical Results (Software Class)

5) Software Status

Level Three (Program): The software development productivity for operational software was statistically equal to that for non-operational software for the program-level. However, there was a possible mapping problem concerning this field in the NCCA Raw Database. Because this classification was not typically reflected in the source databases, NCCA was required to subjectively map programs into the operational or non-operational categories. Therefore, the possibility exists that some of the data points were incorrectly mapped. Due to the uncertainty involved in the actual classification of the data points, a conclusive determination could not be made as to whether software status is a significant productivity driver for the program-level.

Level Four (CSCI): The software development productivity for operational software was statistically equal to that for non-operational software for the CSCI-level. However, this result may be due to other differences in the underlying data sets. The operational data set had many Ada CSCIs and was composed of 61 percent new code and 34 percent verbatim code, on average. The non-operational data set had no Ada CSCIs and was composed of 88 percent new code, on average (no verbatim code). This difference in the composition of the code condition may have increased the average productivity of the operational data set, and decreased the average productivity of the non-operational data set.

To determine whether the results were actually a reflection of differences in code condition and not software status, the test was conducted again with 100 percent new CSCIs. This time, the productivity to develop operational software was statistically lower than that to develop non-operational software. However, this result also may have been skewed by additional differences in the underlying data sets.

To determine whether the programming language, Ada in particular, was driving the productivities rather than software status, the two data sets were tested again with all Ada data points eliminated from the data sets. The two data sets were statistically equal.

In addition to these inconclusive results, the possibility exists that some of the data points were incorrectly mapped, as in the Level Three analysis. Due to the uncertainty involved in the actual classification of the data points, as well as the inconclusive results, a valid determination could not be made as to whether software status is a significant productivity driver for the CSCI-level. Table 4-17 provides a summary of results and corresponding statistics for both Level Three and Level Four software status tests.

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|---------|-------------------|---------|-------------|------|--------------|--------|
| Program | Operational | 0.22 | 1.013 | 50% | Mann-Whitney | Yes |
| | Non-Op | 0.22 | 0.849 | 39% | | |
| CSCI | Operational | 0 | 2.182 | 86% | Mann-Whitney | Yes |
| | Non-Op | 0 | 1.480 | 81% | | |
| CSCI | Op (100% New) | None | 3.271 | 86% | Mann-Whitney | No |
| | Non-Op (100% New) | None | 1.235 | 106% | | |
| CSCI | Op (Non-Ada) | 0 | 2.985 | 83% | Mann-Whitney | Yes |
| | Non-Op (Non-Ada) | 0 | 1.480 | 81% | | |

Table 4-17: Level Three and Level Four Statistical Results (Software Status)

6) Software Mode

Level Three (Program): The software development productivity for embedded mode software was statistically lower than that for non-embedded mode software. For the program-level, software mode is a productivity driver and should be considered when estimating productivity.

Level Four (CSCI): The software development productivity for embedded mode software was statistically lower than that for non-embedded mode software for the CSCI-level. This was consistent with the Level Three (program-level) result. Therefore, software mode is a productivity driver and should be considered when estimating productivity.

Table 4-18 provides a summary of results and corresponding statistics for both the Level Three and Level Four software mode tests.

| Level | Metric | Efactor | Hours/ESLOC | CV | Test | Equal? |
|---------|------------------------------|---------|-------------|-----|--------------|--------|
| Program | Embedded versus Non-Embedded | 0.15 | 2.223 | 41% | Mann-Whitney | No |
| | | 0.09 | 1.322 | 71% | | |
| CSCI | Embedded versus Non-Embedded | 0 | 2.250 | 86% | Mann-Whitney | No |
| | | 0.36 | 0.846 | 47% | | |

Table 4-18: Level Three and Level Four Statistical Results (Software Mode)

7) Language

Level Three (Program): The software development productivity for programs written in Ada was statistically equal to that for programs written in another HOL. The two data sets were statistically equal despite the finding that the average productivity of Ada programs was twice as high as non-Ada programs (0.977 Hours/ESLOC for Ada and 1.807 Hours/ESLOC for non-Ada). However, this finding should be viewed with caution. The large difference in average productivities could be attributed to other differences in the underlying data sets. The Ada data set was composed entirely of support software, 100 percent HOL, non-embedded mode, and simulation programs. In addition, the Ada data set consisted on average of only 36 percent new code, 28 percent modified code, and 36 percent verbatim code. All of these attributes may have increased the average productivity of the Ada data set. The non-Ada data set was composed of C³ mission, radar and simulation programs, which were primarily non-embedded mode. The non-Ada data set was composed on average of 71 percent new code and only seven percent verbatim code, which also may have contributed to a lower average productivity for this data set. Based on these results, a defensible conclusion could not be made as to whether Ada was a productivity driver for the program-level.

Level Four (CSCI): The software development productivity for CSCIs written in Ada was statistically higher than that for CSCIs written in some other HOL. However, there is uncertainty in this result due to differences in the underlying data sets. The Ada data set consisted almost entirely of embedded mode, anti-submarine warfare, and application CSCIs, while the non-ADA data set was primarily embedded mode with a C³ mission. Based on these results, a valid conclusion as to whether Ada was a productivity driver for the CSCI-level cannot be made.

Table 4-19 provides a summary of results and corresponding statistics for both the Level Three and Level Four language tests.

Section 4 – Effort Analysis: Significant Drivers

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|---------|-----------|---------|-------------|-----|--------------|--------|
| Program | Ada | 0.3 | 0.977 | 29% | Mann-Whitney | Yes |
| | Non-Ada | 0.03 | 1.807 | 67% | | |
| CSCI | Ada | 0.02 | 1.512 | 55% | t-test | No |
| | Non-Ada | 0 | 2.626 | 88% | | |

Table 4-19: Level Three and Level Four Statistical Results (Language)

8) Size

Level Three (Program): The software development productivity for small programs was statistically equal to that for large programs. For the program-level, size was not a productivity driver and does not need to be considered when estimating software development productivity.

Level Four (CSCI): The software development productivity for small CSCIs was statistically equal to large CSCIs. This was consistent with Level Three (program-level) results. Therefore, program size was not a productivity driver and does not need to be considered when estimating software development productivity.

Although software size does not appear to be a productivity driver, there may possibly be a critical size value (other than the mean) which will be explored in Section 5 - **Effort Analysis: Normalized Regressions**.

Table 4-20 provides a summary of results and corresponding statistics for both Level Three and Level Four software size tests.

| Level | Attribute | Efactor | Hours/ESLOC | CV | Test | Equal? |
|---------|-----------|---------|-------------|-----|--------------|--------|
| Program | Small | 0.07 | 1.630 | 48% | Mann-Whitney | Yes |
| | Large | 0.07 | 1.857 | 72% | | |
| CSCI | Small | 0 | 2.124 | 83% | t-test | Yes |
| | Large | 0 | 2.066 | 91% | | |

Table 4-20: Level Three and Level Four Statistical Results (Size)

4.4 CONCLUSIONS

This section discusses those attributes NCCA identified as productivity drivers. The decision as to whether an attribute is a driver was based on both statistical test results and on the analyses of the underlying data sets.

Based on the Level One analysis, the following attributes are statistically significant productivity drivers and should be considered when estimating software development productivity.

- 1) Mission (MIS versus Weapon System)
- 2) Counting Convention (Physical versus Logical)
- 3) Language (Assembly versus HOL)
- 4) Phasing (SSR through FQT versus SDR through FQT)

Since the productivities of CSCI-level, program-level, and one-CSCI-level programs were statistically equal, there were no productivity drivers identified from the Level Two analysis. Table 4-21 is a summary of the Level Three and Level Four results. A “Yes” indicates the attribute was a productivity driver, a “No” indicates the attribute was not a productivity driver,

Section 4 – Effort Analysis: Significant Drivers

and a “?” indicates that a defensible conclusion could not be made as to whether or not the metric was a productivity driver. NCCA’s final criterion for concluding that an attribute was a productivity driver was significance at both the program- and CSCI-levels. The attributes that satisfied this criterion are indicated by a checkmark (✓).

| Attribute | Level 3 Analysis (Program-Level) | Level 4 Analysis (CSCI-Level) | Significant at Both Levels? |
|----------------|-------------------------------------|----------------------------------|-----------------------------|
| Code Condition | Yes | Yes | ✓ |
| Platform | Insufficient Data | ? | |
| Mission Area | ? | ? | |
| SW Class | Insufficient Data | Yes | |
| SW Status | ? | ? | |
| SW Mode | Yes | Yes | ✓ |
| Language | ? | ? | |
| Size | No | No | |

Table 4-21: Level Three and Level Four Statistical Results (Summary)

In summary, at a minimum, the analyst should determine the domain or higher-level mission (MIS versus weapon systems), counting convention, language, phasing, code condition, and software mode of the program to be estimated. Due to lack of data, a defensible conclusion could not be made as to whether any of the other attributes are productivity drivers. Although the other attributes were not definitively identified as productivity drivers, to support future analytical efforts, an attempt should also be made to determine the platform type, mission area, software class, and software status of the program being estimated.

4.5 WEAKNESSES

The main weakness of this analysis was the lack of data. Of the 457 data points in the NCCA Raw Database, only 185 were fully defined (i.e., all attributes used in this analysis were defined in the database) and only 95 of those fully defined data points were normalized (i.e., weapon systems, logical, greater than 70 percent HOL, SDR through FQT, code condition and mode known). Therefore, many data points could not be used in the analysis simply due to incomplete information. This limitation was even more pronounced at the program-level, where only 23 of the 31 normalized data points were completely defined and useable in the entire Level Three analysis.

Another weakness of this analysis was the quality of data in the underlying data sets. Because these programs and the database were not developed in a controlled environment, NCCA was often unable to completely isolate the productivity drivers. Every attempt was made to identify other possible drivers; however, with so many holes in the data sets, it was often impossible to do.

4.6 FUTURE EFFORTS

In the future, NCCA plans to collect more data to enhance the analysis. The additional data will be used to substantiate results from all four levels of the analysis, and to further investigate areas of uncertainty. Additional data will also be used to complete the analysis for those areas that were not investigated due to insufficient data. An effort will be made to adequately define

Section 4 – Effort Analysis: Significant Drivers

all new data so that the underlying data sets can be completely understood and any other possible drivers identified.

NCCA was unable to **completely** isolate an attribute to determine whether it was a productivity driver (e.g., Ada versus platform versus contractor). It was impossible to determine whether two or more attributes were mutually independent or dependent on each other. In the future, NCCA also plans to perform multivariate analyses on the data to determine relationships between software attributes with respect to productivity.

5

EFFORT ANALYSIS: NORMALIZED REGRESSIONS

5.1 INTRODUCTION

Based on the conclusions of the previous section, NCCA normalized the raw database in order to conduct regression analyses. The product of these analyses is a set of standard effort estimating relationships intended for use if, and only if, the analyst is unable to collect contractor-specific data relevant to the software development effort being estimated.

Examples of cases where it is appropriate to use the standard relationships are: 1) if the name of the future software development contractor is unknown and 2) if the program being estimated is so early in development that program requirements are ill defined and therefore qualified contractors have yet to be identified.

This section of the handbook serves three purposes: 1) to summarize the NCCA Raw Database in its normalized form; 2) to discuss the analytical approach used to produce the estimating relationships; and 3) to present the estimating relationships and their application rules. This section of the handbook is comprised of the following subsections:

- Review of the NCCA Normalized Software Effort Database
- Partitioning the Data
- Analytical Approach
- Regression Results
- Evaluation of Program-Level versus CSCI-Level Regressions
- Recommendations
- Conclusions
- Future Efforts

5.2 REVIEW OF THE NCCA NORMALIZED SOFTWARE EFFORT DATABASE

As previously discussed, NCCA used the productivity drivers identified in Section 4 - **Effort Analysis: Significant Drivers** to filter the NCCA Raw Database into a normalized database, hereafter referred to as the NCCA Normalized Database. Table 5-1 again shows how the number of data points diminishes as each normalization criterion is applied.

The normalization process eliminated a significant number of program- and CSCI-level data points. For the program-level, only 31 of 151 data points remain. With so few points, it was difficult to find meaningful subsets of data. A fairly large number of CSCI-level data points still remain, but the next section will show how the CSCI data points are concentrated into specific areas.

Section 5 - Effort Analysis: Normalized Regressions

| | Initial Number of Data Points Program | CSCI |
|---------------------------------------|--|------------------------|
| Start: Top-Level | 151 | 329 ³⁶ |
| Normalizing Factors | Number of Data Points Remaining | |
| | Program | CSCI |
| Mission = Weapon System | 105 | 236 |
| Code Count = Logical | 56 | 185 |
| HOL ≥ 70% | 47 | 146 |
| Scope of Effort = SDR through FQT | 32 | 100 |
| Code Condition Known | 31 | 97 |
| Development Mode Known | 31 | 97 |
| Hours/man-month = known | 31 | 97 |
| Final NCCA Normalized Database | 31 | 97³⁷ |

Table 5-1: Arriving at the NCCA Normalized Database

The final program-level NCCA Normalized Software Effort Database consists of 31 data points. The start dates were not provided for all data points. However, the start dates provided were from 1972 through 1984. These software developments were written in FORTRAN, Ada, and JOVIAL. The SLOC range is from 9 to 1,113 KSLOC. The total effort ranged from 9 to 10,976 man-months. A majority of the program-level data points are semi-detached, while some embedded and organic modes are represented. This database includes various missions, such as: radar, command, control and communications (C³), and simulation, which were installed on both ground and ship platforms. The following are the strengths associated with the program-level database: 1) SEL data points reflect impacts of continuous process improvements; 2) Ada programs are well represented; 3) all development modes are well represented; 4) the size range is robust; and 5) the code condition is robust (modified and verbatim code well represented). The following are the weaknesses associated with the program-level database: 1) only three database sources are represented (Implication - database robustness may be compromised); 2) there are only 31 data points remaining after the normalization process; 3) the data points are primarily old; 4) blank fields in the database (# of CSCIs, start dates ...) prevent the application of innovative techniques; and 5) many applications are missing (missiles, sonars, etc.).

The final CSCI-level NCCA Normalized Software Effort Database consists of 97 data points. Similar to the program-level database, the start dates were not provided for all data points. However, the CSCI start dates provided were from 1972 through 1991. These software developments were written in FORTRAN, Ada, CMS-2, JOVIAL, ATLAS and C. The SLOC range is from 0.411 to 492 KSLOC. The total effort ranged from 2.1 to 5,007 man-months. A majority of the CSCI-level data points are embedded, while some semi-detached and organic modes are represented. This database includes various missions, such as: radar, Anti-

³⁶The analysis in Section 4 – **Effort Analysis: Significant Drivers** proved that one-CSCI programs can be included with CSCI-level data points; therefore, the initial CSCI-level data set includes 23 one-CSCI data points.

³⁷ The analysis in Section 4 – **Effort Analysis: Significant Drivers** proved that one-CSCI programs can be included with CSCI-level data points; therefore, the final normalized CSCI-level data set includes 4 one-CSCI data points.

Section 5 - Effort Analysis: Normalized Regressions

Submarine Warfare (ASW), C³, simulation and missile, which were installed on ground, air and ship platforms. The following are the strengths associated with the CSCI-level database: 1) the total number of data points remaining, after the normalization, are 97; 2) new processes are represented; 3) five database sources are represented; 4) Ada is well represented; and 5) the size range is robust. The following are the weaknesses associated with the CSCI-level database: 1) a large amount of the data points are from one program; 2) 46 of 50 Ada data points are from one program; 3) code condition is primarily new and verbatim (limited modified code); and 5) the number of programs is unknown (SMC doesn't link CSCIs to Programs).

The normalization process eliminated a significant number of the source databases. Entire databases were eliminated because of one or two key differences. Table 5-2 shows which source databases remained after normalization and why the others were deleted.

There are only three source databases included in the NCCA Normalized Database at the program-level, and only five source databases included at the CSCI-level. SMC was not well represented at the program-level, while the MITRE Non-Ada Database and Navy Internal data overwhelmed the other sources at the CSCI-level.

| Source Database | Code | Number of Data Points | | Reason for Database Exclusion |
|---------------------|------|-----------------------|------|--|
| | | Program | CSCI | |
| MITRE Non-Ada | 1 | 13 | 38 | |
| MITRE Ada | 2 | 0 | 0 | Effort does not reflect SDR to FQT |
| SMC | 3 | 4 | 6 | |
| NASA SEL | 4 | 14 | 4 | |
| Navy Internal | 5 | 0 | 45 | Program-level data points utilized physical SLOC counting convention |
| Silver SASET | 6 | 0 | 0 | Counting convention, hours/man-month, and scope of effort unknown |
| REVIC Recalibration | 7 | 0 | 0 | Hours/man-month for data points from non-SMC sources could not be verified |
| IITRI | 8 | 0 | 4 | Did not contain program-level data |

Table 5-2: Normalized Database by Source Database

The remaining source databases were not homogeneous. As Table 5-3 shows, each source database had its own concentration of characteristics. For instance, most of the C³ applications came from the MITRE Non-Ada Database. Most databases focused on areas of the sponsor's interest and likely reflected a set of software developers specific to those areas of interest. In most cases NCCA did not have the original developer's name, and, thus, could not determine whether a source database represented a set of diverse contractors.

| Source Database | Mission | Language | Age | Mode | Other Comment |
|------------------|-------------------------------|---------------------------|-----------------------|--------------------------------------|---|
| 1. MITRE Non-Ada | C ³ , Radar | Fortran, Jovial, CMS-2, C | 10 to 25 yrs | Embedded Semi-Detached | |
| 3. SMC | C ³ , MIS, Missile | Ada, Fortran, Jovial, C | 3 to 17 yrs | Embedded Semi-Detached Organic | |
| 4. NASA SEL | SIM | Fortran, Ada | 6 to 12 yrs | Semi-Detached Organic | Non-DoD Ada data points are program-level |
| 5. Navy Internal | ASW | Ada, CMS-2 | < 8 yrs | Embedded | Ada data points are CSCI-level |
| 8. IITRI | C ³ | Ada | > 7 yrs ³⁸ | Embedded | |

Table 5-3: Key Aspects of Remaining Source Databases

³⁸Although dates were not given, the IITRI report was published in 1989. Therefore, since no data point could have started after 1989, the data is at least seven years old.

5.3 PARTITIONING THE DATA

After the NCCA Normalized Database was created, the next step was to partition and examine the data. With 73 fields for each record in the NCCA Raw Database, a tremendous number of partitions could be made. However, some partitions were either not possible or less useful and/or more subjective than others. For instance, because it is widely believed that developer capability is a productivity driver, it was desirable to partition the data by software developer (i.e., contractor). Unfortunately, most of the data did not provide the software developer's name. If the data did not support a partition at the program-level, it was also not partitioned at the CSCI-level, and vice versa. Thus, consistency was maintained between the program-level and CSCI-level analyses.

Based on a preliminary analysis of the data and a statistical test of the means of the productivity metrics (as documented in Section 4 - **Effort Analysis: Significant Drivers**), NCCA generated seven partitions of the data: one top-level and six lower-levels, as listed below:

- 1) **Top-Level**
- 2) **100 Percent New**
- 3) **Not 100 Percent New**
- 4) **Embedded**
- 5) **greater than 75 percent reuse;**
- 6) **greater than 50 percent reuse; and**
- 7) **greater than zero and less than or equal to 50 percent reuse.**

The first four partitions are a direct result of the significant driver analyses, while the last three are based on current software literature and research. The SEL analysis [10] indicates there is a critical point where the savings due to reuse become significant. Barry Boehm's revised COCOMO model (COCOMO II) also addresses this issue. Therefore, NCCA subjectively defined the last three partitions. Table 5-4 provides summary information about the seven partitions.

| Partition | Number of Data Points | |
|-----------------|-----------------------|------|
| | Program | CSCI |
| Top-level | 31 | 97 |
| 100% New | 8 | 32 |
| Not 100% New | 23 | 65 |
| Embedded | 11 | 80 |
| Reuse > 75% | 8 | 10 |
| Reuse > 50% | 9 | 39 |
| 0 < Reuse ≤ 50% | 14 | 26 |

Table 5-4: Summary of Data Partitions

5.4 ANALYTICAL APPROACH

Section 5 - Effort Analysis: Normalized Regressions

Almost all software effort estimating models start with a core estimating equation, usually a function of the size of the software. The traditional form of the equation is a non-linear relationship of the form:

$$\text{Effort} = a * \text{Size}^b$$

where a is some constant, b is the exponent, size is expressed in SLOC or ESLOC, and effort is expressed in man-months or man-hours. Historically, the exponent b ranged from 0.8 to 1.4. If the exponent is greater than one, as the size of the software increases, the associated effort also increases (i.e., the next line of code will be more expensive than the previous line of code). This effect is known as a diseconomy of scale. If the exponent is less than one, the opposite is true (i.e., the next line of code will be cheaper than the previous line of code), implying an economy of scale. There are varying opinions concerning whether software can ever truly enjoy economies of scale. One reason cited for diseconomies of scale is that as size increases, the complexity of the software increases. However, modern software development practices stress modularization. Thus, while the whole may be complex, each piece, will be less complex.

Reference [18] suggests that team dynamics also play a role. As the size of a team grows, more time is spent communicating along an increasing number of communication paths among team members, resulting in less time for developing software. The effort associated with communication could grow faster than the associated gain in productivity by adding staff (a diminishing return). On the other hand, as more contractors use Ada to develop software, experts agree that the increased usage of reused code should positively affect productivity. Thus, if the program is developed to promote reuse of large portions of code, the program may experience economies of scale.

Two forms of simple least squares regression were performed for each data partition. The first form was as follows:

$$\text{Estimated Effort} = a * [\text{New SLOC} + (\text{Efactor} * \text{Reused SLOC})]^b$$

where Efactor is a value between zero and one. The second form of the regression was:

$$\text{Estimated Effort} = a * [\text{New SLOC} + (\text{Efactor}_1 * \text{Modified SLOC}) + (\text{Efactor}_2 * \text{Other SLOC})]^b$$

Both of these equations use a log-log transformation. The difference between the two equations is in the fidelity of the reused SLOC. In the first case, all SLOC that are reused are grouped together; hence, they are equally weighted with the same Efactor . The equation with two Efactors is more sensitive because it treats modified code separately from the other forms of reused code (i.e., rehosted, translated, verbatim, etc.). This is based on the assertion that modified SLOC would require more effort than other reused code types.

Similar to the methodology used in Section 4 - **Effort Analysis: Significant Drivers**, the Efactors are iteratively derived using a special regression spreadsheet model. The model performs a "tradeoff analysis". During the analysis, the model assigns the Efactor(s) a value between zero and one, and solves for the x variable, ESLOC, for each point in the partition.

Section 5 - Effort Analysis: Normalized Regressions

When the regression is performed on this calculated ESLOC, the standard error and Predict (20)³⁹ are calculated.

The Efactor is then changed and the regression rerun. This continues until the standard error and Predict (20) for all Efactor values, in increments of 0.01 between zero and one, have been computed. The results are then analyzed to determine the value of the Efactor that produced the regression with the lowest standard error. A second analysis is performed to determine which Efactor produced the regression(s) with the highest Predict (20).

Figure 5-1 shows a typical graph produced by the regressions. This graph plots both the standard error curve and the Predict (20) curve. This example is based on ESLOC with one Efactor (i.e., combines all reused SLOC). The regression with the smallest error (left axis) of 0.62 occurred when the Efactor was 0.22. Therefore, with this set of data and this type of regression (one variable), reused SLOC would require 22 percent of the effort that new SLOC required. The 22 percent should be thought of as an average Efactor across all the different types of reused code. This particular example's underlying data contained data points that had modified SLOC and verbatim SLOC. The composition of the reused SLOC has a definite influence on the Efactor.

The smooth line, which looks like a step function in Figure 5-1, is the Predict (20) line. The Efactor that produced the maximum Predict (20) (right axis) of 29 percent was 0.02. Thus, the Efactor (0.22) that gave the minimum standard error and had an associated Predict (20) of 16 percent, was not the same Efactor (.02) that gave the maximum Predict (20).

³⁹Predict (20) is the percentage of time the total residuals are within 20 percent of the actual value. (See Appendix C for more details on Predict (20) calculations.)

Section 5 - Effort Analysis: Normalized Regressions

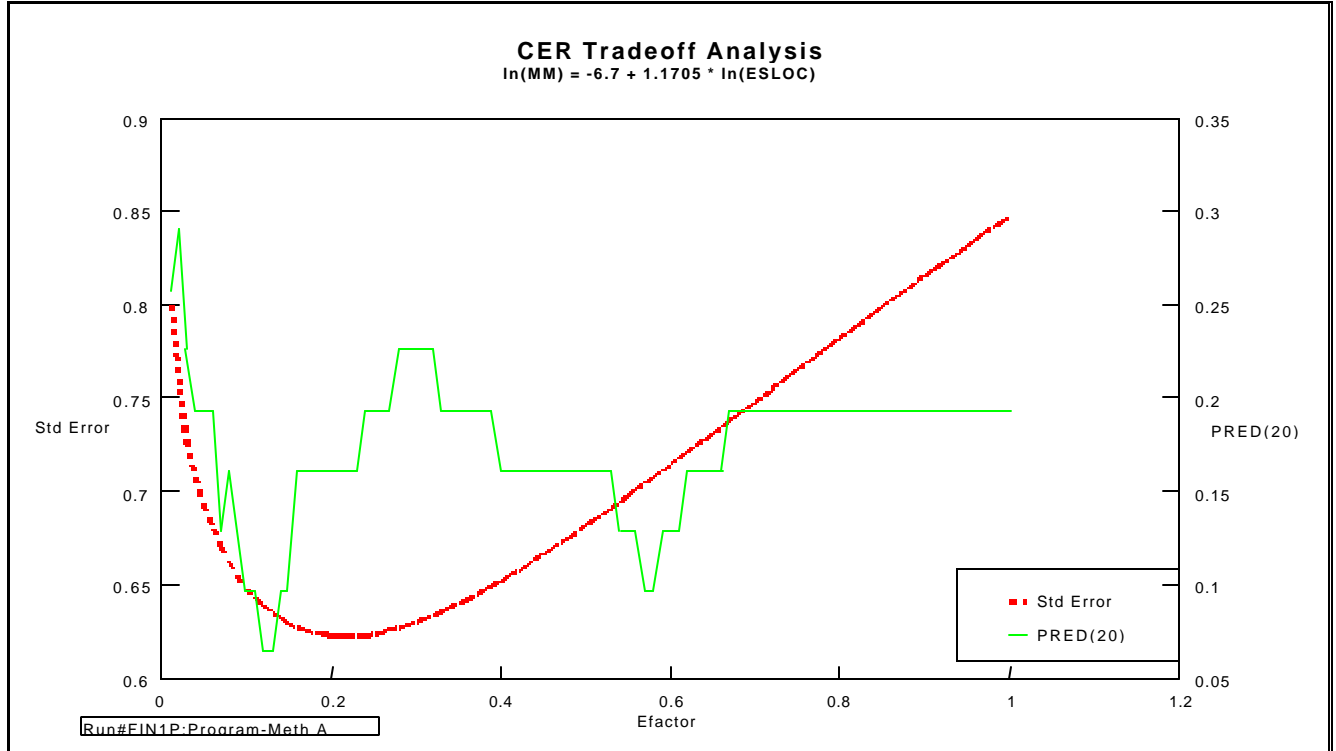


Figure 5-1: Trace of Standard Error and Predict (20)

The standard error for the effort regression with an Efactor of 0.02, was 0.75. Thus, to achieve an increase in Predict (20) from 16 to 29 percent, an increase in the standard error from 0.62 to 0.75 must also occur. In some cases, this tradeoff in error is minimal. In other cases, like this one, the tradeoff is substantial and not desirable. Additionally, after more detailed analysis, it became apparent that to maximize the Predict (20), the statistical tools began trading off on the number of data points within the different database sources. In other words, as shown in Figure 5-2, the data source with the greatest number of data points, which is most likely homogenous, would drive the Predict (20). Due to the database tradeoff involved in the Predict (20) calculations, NCCA preferred to minimize the standard error associated with a regression vice maximizing the Predict (20).

A similar technique was utilized to calculate the minimum standard error and maximum Predict (20) for the regression equations with two Efactors. However, due to the increased number of variables, additional iterations had to be performed.

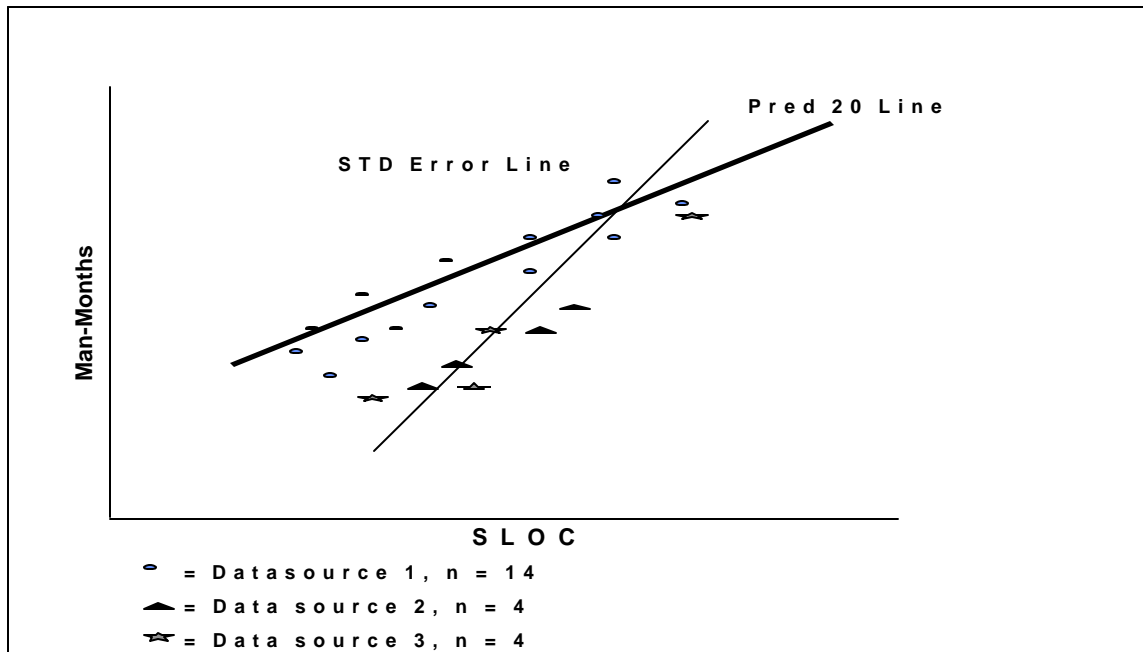


Figure 5-2: Predict (20) Tradeoff

5.5 REGRESSION RESULTS

This section documents the three sets of regressions performed: traditional, non-traditional and revised traditional.

- 1) Traditional regressions express effort (man-months) as a function of ESLOC with dummy slopes or dummy constants. Efactors are used to convert SLOC to ESLOC.
- 2) Non-traditional regressions express effort in two forms: a) effort as a function of new SLOC and reused SLOC; and b) effort as a function of total SLOC and one minus percent reused SLOC (i.e., % new). The second approach discounts effort as more reuse code is used. For these two approaches, Efactors are not used because the actual amount of code or the percent of reused (modified, rehosted, translated, verbatim, etc.) SLOC is an independent variable.
- 3) Revised traditional regressions combine the strengths of the traditional and non-traditional regressions. Effort is a function of ESLOC and % reuse (through the use of an additional dummy variable). This approach also uses Efactors to convert SLOC to ESLOC.

All equations presented below, where effort is in man-months, have been normalized to reflect the 152 hours per man-month standard. Each set, including an analysis of the resulting equations and the elimination process, will be discussed in more detail below.

5.5.1 TRADITIONAL REGRESSIONS

Traditional regressions express effort as a function of ESLOC. Additionally, dummy variables are introduced into the model by way of dummy slopes and dummy constants. By using a dummy variable, a second variable is introduced into the regression. The dummy variable takes the value of zero or one depending on the observation. A dummy constant applies the same impact to all programs, regardless of size, as illustrated in the following equation:

$$\text{Effort} = a * (\text{ESLOC})^b * e^{D_1c}$$

where D_1 is a dummy constant variable.

In contrast, a dummy slope models the non-constant effect attributes have on programs of varying sizes. In other words, a dummy constant applies the same impact (e^{D_1c}) to a program regardless of size, while a dummy slope's impact will change depending upon the size of the program. The net effect of the dummy slope then becomes:

$$\text{Effort} = a * (\text{ESLOC})^{b+D_1c}$$

See Appendix C for a detailed discussion of dummy variables and their application.

For each of the seven data partitions identified earlier, an organized set of regressions was attempted to derive the statistically significant effort regressions at both the program- and CSCI-levels. Table 5-5 summarizes the different scenarios.

| Subset | Number of Data Points Program versus CSCI | Dummy Variables | | |
|------------------------|--|-----------------|----------|---------|
| | | 100% New | Embedded | ESLOC > |
| Top-level | 31 versus 97 | ✓ | ✓ | ✓ |
| 100% New ⁴⁰ | 8 versus 32 | | ✓ | ✓ |
| Not 100% New | 23 versus 65 | | ✓ | ✓ |
| Embedded Only | 11 versus 80 | ✓ | | ✓ |
| Reuse > 75% | 8 versus 10 | | ✓ | ✓ |
| Reuse > 50% | 9 versus 39 | | ✓ | ✓ |
| 0% < Reuse ≤ 50% | 14 versus 26 | | ✓ | ✓ |

Table 5-5: Summary of Regressions

NCCA used dummy slopes in the regressions instead of the more traditional dummy constants because, statistically, the dummy slope outperformed the dummy constant. Additionally, NCCA believes the impact of an attribute (e.g., code condition, mode) varies as the size varies (e.g., the impact of embedded mode may be much greater on a large program than on a small program).

⁴⁰ESLOC calculation is not applicable for this subset since the SLOC are new.

Section 5 - Effort Analysis: Normalized Regressions

As defined in Section 4 - **Effort Analysis: Significant Drivers**, only those variables which were consistently significant at both the CSCI- and program-levels were utilized as dummy variables. Therefore, the attributes represented by the dummy variables applied in the regression runs were similar to those used in partitioning the data. The only dummy variable that was not a criterion for data partitioning was “**ESLOC>?**”. This particular dummy variable was applied to determine whether a critical size value (other than the median which was tested in Section 4 - **Effort Analysis: Significant Drivers**) existed.

A total of 240 candidate traditional regressions were developed. Appendix D details each regression. Some general findings about the regressions are:

- Net exponents (including dummy slopes) for program-level regressions ranged from 0.75 to 1.6, while net exponents for CSCI-level regressions ranged from 0.55 to 1.5.
- Program-level regressions typically had net exponents greater than one, while CSCI-level regressions typically had net exponents less than one. However, the constant for the CSCI-level regressions was usually greater (sometimes by a factor of ten) than that for the program-level regressions.
- CSCI-level regressions exhibited greater variance than program-level regressions, possibly because the CSCI regressions included more database sources, which were non-homogeneous.
- Efactor weights for reused SLOC were greater at the program-level than at the CSCI-level. This is probably due to the underlying composition of the program-level versus CSCI-level databases (i.e., CSCI data consisted of reused code which was primarily verbatim, and therefore, less complex while the program data did not).
- CSCI-level regressions with two Efactors generally exhibited a lower variance than regressions with one Efactor. This effect was the exact opposite for program-level regressions.

Once the 240 regressions were completed, undesirable (biased, inaccurate) regressions were eliminated. Three rounds of elimination were performed to arrive at the final set of acceptable regressions. The first round was very broad, while the last round was performed for very specific reasons. A summary of each round of elimination follows.

5.5.1.1 Round One Eliminations

In this first round, all estimating relationships that did not have significant statistics were eliminated. Significance was set at the 95 percent confidence level. Additionally, some of the effort regressions had the same Efactor when optimized for minimum standard error and maximum Predict (20); that is, the equations duplicated each other. All duplicate equations were also eliminated during round one, leaving a total of 132 candidates.

5.5.1.2 Round Two Eliminations

In this round, all remaining estimating relationships were investigated to determine if the form of the equation (the coefficients and exponents) made technical sense. Several problems were uncovered and are discussed in detail below.

Two Efactors: The set of regressions consisting of two Efactors had some drawbacks. Efactor₁ weighted modified SLOC while Efactor₂ weighted the remaining reused (rehosted, verbatim, or translated) SLOC. In many program-level regressions, the resulting relationships estimated a higher weight for the remaining reused SLOC than for modified SLOC (i.e., Efactor₂ > Efactor₁). This implied that modified SLOC were easier to develop than the remaining reused SLOC. This did not make sense technically. Since most databases did not specifically categorize reused SLOC, NCCA had to develop mapping guidelines to classify them in the NCCA Raw Database. For instance, if a program consisted of zero percent redesign and recode, NCCA mapped the associated SLOC into the verbatim field. It is possible that NCCA's mapping scheme was inaccurate. Although most of the CSCI-level regressions did not exhibit this problem, the rules used to map their SLOC were the same as those used at the program-level. Thus, if there is a mapping problem at the program-level, then there is also a problem at the CSCI-level. Therefore, all regressions of the two Efactor form were eliminated. This procedure eliminated 75 of the remaining 132 candidate equations, thus leaving 57 equations.

Critical ESLOC>?: Significant relationships were developed in which a critical size dummy variable was determined. However, in some regressions, the dummy slope calculation was based on only 2 or 3 data points. In other regressions, the resulting equation weighted the dummy slope counterintuitively. For example, at the CSCI-level, a regression found 10,000 ESLOC to be the critical size value; however, anything smaller than 10,000 ESLOC was estimated to be more expensive while anything greater was estimated to be less. This is conceivable, if there is a given amount of level of effort required regardless of the size of the program. However, at the program-level, the results were reversed (i.e., programs smaller than the critical size value were less expensive). NCCA could not explain the contradiction, therefore, regressions with **ESLOC>?** as a dummy variable were eliminated. This reduced the number of equations by an additional 20, leaving a total of 37 estimating relationships for the next round of elimination.

5.5.1.3 Round Three Eliminations

The third round of elimination focused on program and CSCI specific reasons for exclusion. Three areas were analyzed: 1) performance and comparison of top-level equations versus lower-level equations; 2) robustness of the underlying data utilized in the equation; and 3) degrees of freedom of the resulting equations.

Program-Level: The top-level equation [5-1], which was derived from all 31 data points and had an embedded dummy slope, mimicked its corresponding lower-level equation [5-2], which consisted of only the 11 embedded data points as shown below:

Top-Level Equation (Program-Level, Embedded)

| |
|--|
| $\text{Effort} = 0.0041 * [\text{New SLOC} + (0.19 * \text{Reused SLOC})]^{[1.0377 + (0.0651 * D_i)]}$ |
|--|

[5-1]

Section 5 - Effort Analysis: Normalized Regressions

| | | | |
|---|-------------------------|---|--------------------------|
| $R^2 = 0.92$ | Std Error = 0.54 | Std Error_{embedded}⁴¹ = 0.45 | Predict (20)= 32% |
| Predict (20)_{embedded}⁴¹ = 55% N = 31 | | | |
| where D_1 equals one if the program is embedded and zero otherwise. | | | |

Lower-Level Equation (Program-Level, Embedded)

| | | | |
|--|--|---------------------------|---------------|
| Effort_{embedded} = 0.0041 * [New SLOC + (0.09 * Reused SLOC)]^(1.1101) | | | |
| $R^2 = 0.92$ | Std Error_{embedded} = 0.42 | Predict (20) = 45% | N = 11 |

[5-2]

The constant slope (1.1101) and net slope ($1.0377 + 0.0651 = 1.1028$) of the two equations are almost exactly the same. While the R^2 s are identical, the standard error of 0.42 for equation [5-2], is slightly better than the partial standard error of 0.45 for equation [5-1]; however, there is a significant tradeoff in degrees of freedom (9 versus 28) when using equation [5-2].

The top-level equation [5-3], which was derived from all 31 data points and had a 100 percent new dummy slope, also mimicked its corresponding lower-level equation [5-4], in which only 100 percent new data points were utilized.

The partial standard error in the top-level equation [5-3] is the same as the standard error in the lower-level equation [5-4], yet a very large tradeoff in degrees of freedom again occurred while using the lower-level regression. In contrast, the R^2 and Predict (20) are higher for the lower-level equation [5-4]. However, as discussed previously, due to the source database influences involved in the Predict (20) calculations, NCCA prefers to minimize the standard error associated with a regression. Therefore, since equations [5-1] and [5-3], the top-level equations with corresponding dummy slopes, perform as well as equations [5-2] and [5-4], these lower level equations were eliminated.

Top-Level Equation (Program-Level, 100% New)

| | | | |
|---|------------------------|---|---------------------------|
| Effort = 0.0013 * [New SLOC + (0.35 * Reused SLOC)]^[1.1345 + (0.0841 * D_1)] | | | |
| $R^2 = 0.93$ | Std Error = 0.5 | Std Error_{100% New}⁴² = 0.27 | Predict (20) = 50% |
| Predict (20)_{100% New}⁴² = 55% N = 31 | | | |
| where D_1 equals one if the program is 100% new and zero otherwise. | | | |

[5-3]

Lower-Level Equation (Program-Level, 100% New)

| | | | |
|--|-------------------------|---------------------------|--------------|
| Effort_{100% New} = 0.0011 * (Total)^(1.2304) | | | |
| $R^2 = 0.98$ | Std Error = 0.27 | Predict (20) = 63% | N = 8 |

[5-4]

CSCI-Level: Similar to the program-level regressions, the top-level equation, with an embedded dummy, mimicked the corresponding lower-level equation. Therefore, the lower-level equations from the embedded subsets of data were eliminated.

Program- and CSCI-Level: Table 5-6 shows the distribution of data points for each of the lower-level equations. Many of the lower-level equations that remained had low standard error

⁴¹StdError_{embedded} and Predict (20)_{embedded} represent the standard error and Predict (20) of the equations when applied to the underlying embedded data points (N=11) only, vice calculating the standard error and Predict (20) of the overall equation, which would be calculated using all 31 data points

⁴²StdError_{100% New} and Predict (20)_{100% New} represent the standard error and Predict (20) of the equations when applied to the underlying 100% New data points (N=8) only, vice calculating the standard error and Predict (20) of the overall equation, which would be calculated using all 31 data points

Section 5 - Effort Analysis: Normalized Regressions

and high Predict (20) values; however, the number of data points and, therefore, the degrees of freedom were small. For example, the >75% reuse data set consisted of eight and ten data points at the program- and CSCI-levels, respectively. The small data sets were especially prevalent at the program-level.

| Partition | Level | Source Database Code ⁴³ | | | | | |
|------------------|---------|------------------------------------|---|----|----|---|-------|
| | | 1 | 3 | 4 | 5 | 8 | Total |
| 100% New | Program | 5 | 3 | | | | 8 |
| | CSCI | 23 | 5 | 1 | 0 | 3 | 32 |
| Not 100% New | Program | 7 | 1 | 15 | | | 23 |
| | CSCI | 15 | 1 | 3 | 45 | 1 | 65 |
| Reuse > 75% | Program | 2 | 1 | 5 | | | 8 |
| | CSCI | 6 | | 1 | 3 | | 10 |
| Reuse > 50% | Program | 3 | 1 | 5 | | | 9 |
| | CSCI | 9 | 0 | 1 | 29 | | 39 |
| 0% < Reuse ≤ 50% | Program | 5 | | 9 | | | 14 |
| | CSCI | 6 | 1 | 2 | 16 | 1 | 26 |
| Embedded | Program | 9 | 1 | | | | 10 |
| | CSCI | 29 | 2 | | 45 | 4 | 80 |

Table 5-6: Subset Distribution of Data Points Across Source Databases

Additionally, many of the other lower-level regressions consisted of only two database sources. For example, at the program-level, the embedded partition consisted of only 10 data points and all but one came from the same database source. For these reasons, all of the remaining lower-level program-level equations were eliminated. All of the CSCI-level lower-level equations were also eliminated except for those from the 100% new and not 100% new partitions. These lower level equations did not mimic the top-level equations with associated dummy variables, as was the case at the program-level, and they did not suffer from low degrees of freedom or a low number of database sources.

After three rounds of elimination, a total of 16 equations remained. NCCA next eliminated equations which maximized Predict (20) vice minimizing the standard error, based on the biases previously discussed (i.e., the regressions were fitting the curve through the data source with the greatest number of data points). This left 10 significant equations, eight top-level program- and CSCI-level equations with and without dummy variables and two CSCI lower level equations. Of these, NCCA selected the equations with the lowest standard error that also captured the effects of the significant drivers identified in Section 4 - **Effort Analysis: Significant Drivers**.⁴⁴ This resulted in the following four equations:

Top-Level Equation (Program-Level)

$$\text{Effort} = 0.0028 * [\text{New SLOC} + (0.3 * \text{Reused SLOC})]^{[1.0549 + (0.0668 * D_1 + (0.0427 * D_2))]}$$

R² = 0.94 Std Error = 0.47 Predict (20) = 35% N = 31 Range = 4.2 - 72.3 EKSLOC

where D₁ equals one if the program is 100% new and zero otherwise; and D₂ equals one if the program is embedded and zero otherwise

[5-5]

Top-Level Equation (CSCI-Level)

⁴³ The code number NCCA assigned to each source database: 1 = MITRE Non-Ada; 3 = SMC; 4 = NASA SEL; 5 = Navy Internal; 8 = IITRI Report.

⁴⁴ The mode dummy was insignificant for the not 100 percent new lower-level equation.

Section 5 - Effort Analysis: Normalized Regressions

$$\text{Effort} = 0.0229 * [\text{New SLOC} + (0.03 * \text{Reused SLOC})]^{[0.8609 + (0.0315 * D_1 + (0.0529 * D_2))]} \quad [5-6]$$

$R^2 = 0.77$ Std Error = 0.67 Predict (20) = 26% N = 97 Range = 0.4 - 253.4 EKSLOC

where D_1 equals one if the program is 100% new and zero otherwise; and D_2 equals one if the program is embedded and zero otherwise

Lower-Level Equation (CSCI-Level, 100% New CSCIs)

$$\text{Effort} = 0.0387 * (\text{Total SLOC})^{[0.779 + (0.1269 * D_1)]} \quad [5-7]$$

$R^2 = 0.76$ Std Error = 0.8 Predict (20) = 22% N = 32 Range = 0.4 - 128.2 KSLOC

where D_1 equals one if the CSCI is embedded and zero otherwise

Lower-Level Equation (CSCI-Level, Not 100% New CSCIs)⁴⁴

$$\text{Effort} = 0.0114 * [\text{New SLOC} + (0.04 * \text{Reused SLOC})]^{(0.9766)} \quad [5-8]$$

$R^2 = 0.81$ Std Error = 0.56 Predict (20) = 31% N = 65 Range = 1.5 - 255.8 EKSLOC

For the CSCI-level, there remained two possible alternatives to estimate effort, either a top-level equation [5-6] or a set of lower level equations based on whether the CSCI was 100 percent new [5-7] or less than 100 percent new [5-8]. As previously stated, the program-level, lower-level 100 percent new and not 100 percent new equations were eliminated because they mimicked the top-level regressions.

100 PERCENT NEW PROGRAMS

A problem was discovered upon further examination of the 100 percent new CSCI-level equation [5-7]. Figure 5-3 provides a comparison of two hypothetical programs. If program A is a 100 percent new program, then the CSCIs that constitute program A will also be 100 percent new. Program B is not 100 percent new. However, it is definitely possible within a program that is not 100 percent new to have a mixture of CSCIs that are 100 percent new and CSCIs that have some level of reuse.

| Program A (100% New) | | Program B (< 100% New) | |
|----------------------|----------|------------------------|----------|
| CSCI #1A | 100% New | CSCI #1B | 100% New |
| CSCI #2A | 100% New | CSCI #2B | 80% New |
| CSCI #3A | 100% New | CSCI #3B | 50% New |
| Total Effort | EA | Total Effort | EB |

Figure 5-3: One Hundred Percent New CSCIs

When the data was filtered for 100 percent new CSCIs, CSCIs from both 100 percent new and less than 100 percent new programs would be included. Thus, the 100 percent new CSCIs from 100 percent new programs would be combined with the 100 percent new CSCIs from not 100 percent new programs. Once this was accomplished, however, it appeared that the productivity metrics associated with CSCIs #1A, #2A, and #3A were worse than the productivity of CSCI #1B.

A subset of the NCCA Normalized Database was used to determine if there was a difference between 100 percent new CSCIs from 100 percent new programs and 100 percent new CSCIs from not 100 percent new programs. The average hours per SLOC for 100 percent new CSCIs from 100 percent new programs was twice as high as the average hours per SLOC for the 100 percent new CSCIs from not 100 percent new programs.

Section 5 - Effort Analysis: Normalized Regressions

Intuitively, it makes sense that CSCI #1B would be cheaper per line of code than any corresponding CSCI in Program A. The effort to design a brand new CSCI for an existing system is probably easier than designing a brand new CSCI for a system which does not exist yet. One possible explanation for this difference is:

Improper Effort Allocation: The effort data for some programs in the NCCA Raw Database appears to have been allocated to the CSCI-level. If effort was allocated, it was probably done on a pro rata basis (bigger CSCIs get more effort allocated to them, smaller CSCIs get less). A 100 percent new CSCI would have more requirements analysis than a not 100 percent new CSCI. However, if the effort was allocated from the program-level, the 100 percent new CSCI may receive only a proportion (based on size) of the effort associated with requirements analysis instead of its true share. The result would be that the 100 percent new CSCI from the not 100 percent new program would erroneously appear to require less effort than a 100 percent new CSCI from a 100 percent new program (which would not suffer this allocation problem).

Ultimately, the reasons for the differences experienced are unknown. If there is an allocation problem, it can only be detected and corrected during future collection of historical data.

- **Based on this discrepancy, NCCA recommends that the analyst not estimate a 100 percent new program with the sum of 100 percent new CSCI-level regressions.**

In theory, this also means that a CSCI-level regression should not mix CSCIs from 100 percent new programs with those from not 100 percent new programs. The original 100 percent new CSCI-level regression (resulting in equation [5-7]) included this mixture. Unfortunately, due to the insufficient number of data points, it was not possible to develop separate regressions.

Again, based on the above discussion of the difference in 100 percent new CSCIs, NCCA recommends that 100 percent new programs be estimated with the top-level program equation [5-5] vice estimated at the CSCI-level with equation [5-6] and then summed. As a result, equations [5-7] and [5-8] were eliminated, leaving one top-level program regression (equation [5-5]) and one top-level CSCI regression (equation [5-6]). These regressions are in Appendix D.

Overall, the strengths of these equations are: 1) they quantitatively solve for the Efactor, so the uncertainty and variance of the Efactor are reflected in the overall equation's statistical results; and 2) they account for the significant drivers (i.e., code condition and mode) without sacrificing degrees of freedom. However, the weaknesses are:

- 1) The regressions do not account for high reuse programs separately (i.e., high reuse programs and CSCIs are averaged into the regressions along with low reuse programs and CSCIs and thereby drive the resulting average productivity up). Therefore, the regressions probably overestimate productivity on programs or CSCIs with low reuse that are not 100 percent new (100% new programs or CSCIs are accounted for with a dummy variable) and underestimate productivity of high reuse programs.
- 2) The program-level regression's [5-5] underlying database consists primarily of non-embedded data points, so resulting productivity metrics may be optimistic when applied to

Section 5 - Effort Analysis: Normalized Regressions

programs which are primarily embedded. However, the application of the dummy variable attempts to account for the embedded programs, and the residuals do not indicate any bias.

- 3) The CSCI-level regression [5-6] is driven by one underlying program with 45 CSCIs (i.e., 46 percent of the database); therefore, if programs estimated are significantly different from this program, the regression may not be appropriate.
- 4) The CSCI-level regression [5-6] exponent is less than one, which implies economies of scale (as the CSCI size increases, productivity improves). As discussed on page 5-4, there are varying opinions on the feasibility of economies of scale. In practice, NCCA expects that as the program size increases, the number of CSCIs will also increase. Therefore, since the CSCI-level traditional equation is applied at the CSCI-level, the constant of the equation will be reapplied for each additional CSCI, which in effect eventually negates the impact of the exponent. In other words, the larger the program (SLOC), typically, the larger the CSCI count, and hence, the more times the constant will be added to the overall estimate. At some point (depending on the size of and number of CSCIs associated with the program), the CSCI equation crosses over and actually estimates more effort than the program-level equation, which has a smaller constant but an exponent greater than one (i.e., economies of scale are no longer realized).

Despite these weaknesses, the regressions are still valid approximations, and they are considered in the final analysis (Section 5.7), which compares the remaining viable regressions.

5.5.2 NON-TRADITIONAL REGRESSIONS: SET ONE

Thus far, the focus of the discussion has centered around a fairly traditional regression for software effort of the form:

$$\text{Actual Effort} = k (\text{ESLOC})^n$$

In addition to the final two traditional candidates (equations [5-5] and [5-6]), a set of "non-traditional" effort regressions was developed. This analysis was based on developing regressions where effort was a function of SLOC without the use of an Efactor. Two different sets of analyses were developed and are detailed below.

The first approach was to specifically filter the database for all data points that have the same type of SLOC. For example, one filter was for data points with non-zero values for new and modified SLOC. If enough data points were present, a direct regression (without the need for Efactors) was performed.

NCCA created several subsets of data that met this criteria. The resulting regression equations are outlined below:

$$\text{Actual Effort} = a * (\text{New SLOC})^b * (\text{Modified SLOC})^c$$

$$\text{Actual Effort} = a * (\text{New SLOC})^b * (\text{Verbatim SLOC})^c$$

$$\text{Actual Effort} = a * (\text{New SLOC})^b * (\text{Reused SLOC})^c$$

Section 5 - Effort Analysis: Normalized Regressions

These regressions were performed at both the program- and CSCI-levels. The equations that contained modified SLOC as an independent variable were significant for both the CSCI- and program-levels. The equations that contained verbatim SLOC as an independent variable were not significant in either case (program- or CSCI-level). This makes sense because the Efactor weights that were calculated for the verbatim code in the traditional regressions were quite small (between 0.01 and 0.03). Unless the regression statistics were very tight, this small weight would be insignificant and difficult to obtain. The regressions based on reused SLOC had mixed results. They were significant at the program-level, but not at the CSCI-level. Again, this was due to the underlying database. The program-level reused SLOC consisted of a large amount of modified SLOC, while the CSCI-level data points consisted primarily of verbatim SLOC.

Overall, the strengths of these equations are: 1) they do not require an Efactor, so one level of uncertainty and variance is reduced and 2) they resulted in better statistics at the CSCI-level. However, the weaknesses are: 1) the underlying database sizes are small; 2) the issue of mapping the reused SLOC into the correct category still remains; 3) there were not enough data points to develop regressions for other types of reused code, such as translated and rehosted; 4) they resulted in worse statistics at the program-level, and 5) it is not clear how dummy slopes would be handled with this type of model (i.e., are they applied to new SLOC, modified SLOC, or both?). Because of these weaknesses, NCCA eliminated this set of non-traditional regressions from further consideration.

5.5.3 NON-TRADITIONAL REGRESSIONS: SET TWO

A second type of non-traditional effort regression was also investigated:

$$\text{Effort} = a * (\text{Total SLOC})^b * (1 - \% \text{Reused SLOC})^c$$

The first term, $a * (\text{Total SLOC})^b$, is defined as a nominal effort scalar. The second term, $(1 - \% \text{Reused SLOC})^c$, is defined as a reuse discount factor. When regressions are performed, c is a value between zero and one. Therefore, as the amount of reuse increases from zero to 100 percent, the term $(1 - \% \text{Reused})^c$ decreases. When the amount of reuse is very low, the estimated effort will essentially be derived from the first part of the equation. As the amount of reuse increases, an increasing amount of effort will be removed from the nominal effort scalar value. The results of the regressions are provided below as well as in Appendix D:

Program-Level Equation

| | | | | | |
|--|-------------------------|---------------------------|---------------|----------------------------------|-------|
| Effort = 0.0015 * (Total SLOC)^(1.1075) * (1 - %Reused SLOC)^(0.3329) | | | | | [5-9] |
| R² = 0.91 | Std Error = 0.52 | Predict (20) = 18% | N = 22 | Range 9.0 - 1,113.0 KSLOC | |

CSCI-Level Equation

| | | | | | |
|--|-------------------------|---------------------------|---------------|--------------------------------|--------|
| Effort = 0.0108 * (Total SLOC)^(0.9767) * (1 - %Reused SLOC)^(0.8394) | | | | | [5-10] |
| R² = 0.81 | Std Error = 0.56 | Predict (20) = 32% | N = 65 | Range 4.7 - 492.0 KSLOC | |

The major strength of this method is that it introduces the notion of a non-linear discount factor for reuse (i.e., the proportionate amount of effort removed from the nominal effort is much greater at a reuse level of 50 percent than it would be at a reuse level of 10 percent). Figure 5-4 demonstrates this graphically.

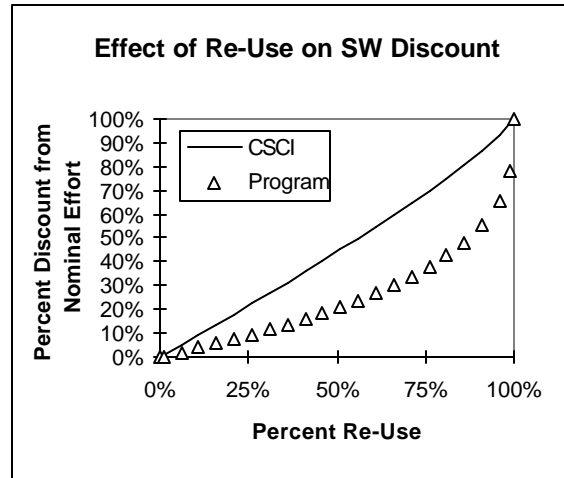


Figure 5-4: Effort Discount as a Function of Reuse

A more detailed analysis of the discount factor shows that the critical reuse value (the point where the standard error is minimized) at the program-level is 82 percent (i.e., percent reuse becomes significant at greater than or equal to 82 percent). See Appendix D for supporting documentation.

Another strength of this model is that the effect of changing from "new" to "reused" is not a step function, but a smooth curve. In the traditional models with Efactors, changing the amount of reuse by one line of code has the same effect on ESLOC whether it is the first line of reuse or the last, therefore, the same "discount" is always applied (i.e., the Efactor is constant). With the reuse discount factor applied in equations [5-9] and [5-10], a cumulative effect is achieved. The amount of discount for the next line of reused code is greater than the previous line of reused code (i.e., the Efactor is not constant). The final strength was the database size. Both regressions had a robust set of data (included more than one database source) and both had a sufficient sample size.

Weaknesses of this approach are: 1) the standard error is slightly higher at the program-level when compared against the traditional effort regression (equation [5-5]); 2) dummy slopes, such as embedded mode, a statistically proven productivity driver, were not utilized. The dummy slopes could only be applied to one term in the equation (either the lines of code term or the reused term), and it could not be determined which was correct; 3) the regressions yield inaccurate estimates if the percent reused equals 100 percent (i.e., effort = $1 - 1 = 0$). However, NCCA questions the accuracy of any sizing estimate for a program that includes no new development; and 4) the program-level discount factor is exceptionally flat in comparison to the CSCI-level discount factor. Specifically, for a program with 90 percent reuse, the program-level regression results in an estimate of approximately 46 percent of the effort of a 100 percent new program. On the other hand, the CSCI-level regression estimates the same program to require only 14 percent of the effort of a 100 percent new program. Intuitively, the program-level regression appears to be conservative.

Despite these weaknesses, the regressions are promising and intuitively pleasing, and are considered in the final analyses (Section 5-7), which compares the remaining viable regressions.

5.5.4 REVISED TRADITIONAL REGRESSIONS

Based on the strengths and weaknesses of equations [5-5], [5-6], [5-9], and [5-10], NCCA developed one additional set of traditional regressions. NCCA performed regressions which combined the impacts of the non-traditional discount factor into the traditional equations, by incorporating a dummy variable for percent reused SLOC. These equations also take the form:

$$\text{Effort} = a * (\text{ESLOC})^b * e^{D_1, c}$$

The results of the regressions are provided below as well as in Appendix D:

Top-Level Equation (Program-Level)

| | |
|--|--------|
| Effort = 0.0012 * [New SLOC + (1 * Reused SLOC)] ^[1.1067 + (0.0912 * D₁) + (0.0326 * D₂) - (0.0982 * D₃)] R² = 0.96 Std Error = 0.42 Predict (20) = 58% N = 31 Range = 9.0 - 1,113.0 EKSLOC where D ₁ equals one if the program is 100 percent new and zero otherwise; D ₂ equals one if the program is in embedded mode and zero otherwise; and D ₃ equals one if percent reused is greater than or equal to 82 percent and zero otherwise. | [5-11] |
|--|--------|

Top-Level Equation (CSCI-Level)

| | |
|--|--------|
| Effort = 0.0211 * [New SLOC + (0 * Reused SLOC)] ^[0.8590 + (0.0338 * D₁) + (0.0631 * D₂) + (0.0623 * D₃)] R² = 0.78 Std Error = 0.66 Predict (20) = 29% N = 97 Range = 0.4 - 245.8 EKSLOC where D ₁ equals one if the CSCI is 100 percent new and zero otherwise; D ₂ equals one if the CSCI is in embedded mode and zero otherwise; and D ₃ equals one if percent reused is greater than or equal to 75 percent and zero otherwise. | [5-12] |
|--|--------|

The major strength of this method is that it accounts for reuse, as well as the other significant drivers identified previously (i.e., code condition and mode). The statistics of these equations improve in comparison to the previous top-level regressions (equations [5-5] and [5-6]). The major weakness of this approach at the program-level is that the Efactor equals one (i.e., reused SLOC require the same effort as new SLOC). Although NCCA expected the Efactor to increase when the high reuse programs were normalized through the use of a dummy variable, an Efactor of one is not intuitively pleasing. However, some experts say that reusing code requires more effort than development from scratch. Although counterintuitive, it's not infeasible in certain circumstances. Additionally, an Efactor equal to one also introduces an additional problem when attempting to compare productivity metrics between candidate equations, as demonstrated in the following example.

- Based on Equation [5-5] and an Efactor = 0.3 for an embedded program:

If New SLOC = 10,000 and Reused SLOC = 10,000 then ESLOC = 13,000,
 Effort = 91.75 MM; Productivity = 13,000/(91.75 MM * 152 hrs/MM) = 0.93 **ESLOC/hour** or
 = 20,000/(91.75 MM * 152 hrs/MM) = 1.43 **SLOC/hour**

- Based on Equation [5-11] and an Efactor = 1.0 for an embedded program with less than 82 percent reuse code:

If New SLOC = 10,000 and Reused SLOC = 10,000 then ESLOC = 20,000;
 Effort = 95.36 MM; Productivity = 20,000/(95.36 MM * 152 hrs/MM) = 1.38 **ESLOC/hour**, or
 = 20,000/(95.36 MM * 152 hrs/MM) = 1.38 **SLOC/hour**

Based on the metric “ESLOC/hour”, equation [5-11] appears to be more productive (1.38 ESLOC/hour versus 0.93 ESLOC/hour), when in actuality, it is estimating more effort (95.36 versus 91.75 man-months) for the same program. Hence, the analyst must be careful when comparing productivity metrics among regressions with varying equivalent code conversion techniques.

At the CSCI-level, the major weakness of this approach is the counterintuitive impact of high reuse. For CSCIs with greater than 75 percent reuse, the equation adds effort vice deleting it. This contradicts the program-level equation and NCCA’s expectations. Upon further analysis, it became evident that one data point (NCCA-414) was driving this effect. After deletion of this data point, the derived regression resulted in a critical reuse value of 61 percent (vice the 75 percent previously calculated). Also, the resulting equation deletes effort for those data points which are greater than the critical value, vice adding effort as the original equation predicts. However, unless specific, detailed technical or programmatic information supported deletion of data points, NCCA did not delete apparent outliers.

Another weakness of this approach is that the Efactor for this equation at the CSCI-level is zero, which implies that no additional effort is required for reused code. Although the effort may be small, especially if the code used entirely as is (verbatim), NCCA still contends that there is some effort associated with this code required to gain an understanding of it and verify requirements.

However, despite these weaknesses, the program-level regression is appealing, so this set of equations is also considered in the final analyses, which compares the remaining viable regressions.

5.6 EVALUATION OF PROGRAM-LEVEL VERSUS CSCI-LEVEL REGRESSIONS

Three sets of estimating relationships remained to be evaluated. The first set was the top-level traditional regressions ([5-5] and [5-6]) which incorporated empirically developed Efactors. The second set was the non-traditional regressions: Set Two ([5-9] and [5-10]) based on total SLOC and percent reused SLOC. The final set was the revised top-level traditional regressions [5-11] and [5-12] which also incorporated percent reused SLOC. To try to obtain additional information about the individual sets of regressions, NCCA conducted two levels of comparison. The first set of comparisons compared the CSCI versus Program estimate deltas for each set of regressions (i.e., [5-5] versus [5-6] deltas; [5-9] versus [5-10] deltas; [5-11] versus [5-12] deltas). See Appendix D for details. The second set of comparisons, also provided in Appendix D, compared the estimates derived from each regression at both the program- and CSCI-level (i.e., [5-5] versus [5-9] versus [5-11] and [5-6] versus [5-10] versus [5-12]).

A separate set of validation data was constructed to evaluate how the program and sum of the CSCI-level effort regressions compared. The data was not used at this point to compare estimates with actuals because some of the validation data was non-normalized.⁴⁵ The main criterion for accumulating this data was that the program and CSCIs had to be linked. The

⁴⁵ This data is used later to assess the overall estimating methodology, including using non-normalized productivity factors to estimate the non-normalized data points. This is detailed in Section 7 – Effort Analysis: Overall Process.

Section 5 - Effort Analysis: Normalized Regressions

database sources that provided this information were the MITRE Non-Ada, MITRE Ada, SMC, NASA SEL, Navy Internal, and SASET databases. The data represented programs with different quantities of CSCIs and a range of CSCI sizes. The only other criterion was that the SLOC were counted as logical lines.

A total of 22 programs with associated CSCI information were identified. As Table 5-7 shows, most of the validation data points came from the MITRE Non-Ada Database. Also, there were a significant number of 100 percent new programs. Since no comparison was made at this time between estimates and actuals, the sum of CSCI-level estimates using a CSCI-level 100 percent new estimate, could be used on the 100 percent new programs.

| Record # | Program Size | DB Code | CSCI Count | Avg CSCI Size | Mission | Language | %New |
|----------|--------------|---------|------------|---------------|---------|----------|------|
| NCCA-67 | 11.0 | 5 | 2 | 399.6 | UUV | Ada | 100 |
| NCCA-286 | 134.7 | 6 | 6 | 44.6 | Radar | Assembly | 100 |
| NCCA-287 | 57.0 | 6 | 4 | 43.9 | C3 | HOL | 100 |
| NCCA-291 | 58.5 | 4 | 2 | 234.1 | SIM | Fortran | 15 |
| NCCA-315 | 74.8 | 2 | 6 | 9.5 | C3 | Ada | 51 |
| NCCA-347 | 95.1 | 2 | 7 | 14.6 | SIM | Ada | 100 |
| NCCA-367 | 114.4 | 4 | 2 | 231.9 | SIM | Fortran | 87 |
| NCCA-409 | 185.6 | 1 | 4 | 42.0 | Radar | Jovial | 16 |
| NCCA-410 | 204.0 | 1 | 5 | 204.0 | Radar | Jovial | 80 |
| NCCA-411 | 210.0 | 5 | 3 | 284.2 | ASW | C | 19 |
| NCCA-417 | 231.9 | 1 | 12 | 28.6 | C3 | CMS-2 | 100 |
| NCCA-418 | 234.1 | 1 | 7 | 185.6 | Radar | Jovial | 62 |
| NCCA-422 | 254.1 | 3 | 6 | 338.1 | C3 | Ada | 62 |
| NCCA-425 | 263.2 | 6 | 15 | 295.2 | C3 | Assembly | 100 |
| NCCA-427 | 267.9 | 6 | 4 | 47.3 | Radar | HOL | 37 |
| NCCA-428 | 283.5 | 6 | 6 | 84.7 | Radar | HOL | 87 |
| NCCA-431 | 295.2 | 1 | 18 | 37.4 | C3 | Assembly | 100 |
| NCCA-435 | 338.1 | 1 | 2 | 47.5 | C3 | Fortran | 47 |
| NCCA-441 | 422.5 | 5 | 9 | 2.2 | C3 | Ada | 100 |
| NCCA-454 | 1113.0 | 1 | 4 | 22.4 | C3 | Jovial | 50 |
| NCCA-456 | 1420.8 | 5 | 10 | 1,113.0 | C3 | C | 99 |
| NCCA-458 | 1997.9 | 5 | 52 | 84.5 | ASW | Ada | 47 |

Table 5-7: Summary of Validation Database

As Table 5-8 shows, program- and CSCI-level estimates tended to be within 20 to 50 percent of each other, with mean absolute deviations (MADs) of 26.6 percent, 36.7 percent, and 28.8 percent, respectively. This is very good considering the standard error around most of the final regressions was between 40 and 60 percent. See Appendix D for graphical representations of the final six equations and associated supporting spreadsheets and regression comparison tables for both the program- and CSCI-levels.

A few interesting observations from this analysis were noted (as shown in Figures 5-5 to 5-8):

- No trend was found when the differences were compared against CSCI average size or CSCI count (Figures 5-5 and 5-8 respectively).

Section 5 - Effort Analysis: Normalized Regressions

| | % | EQ #5 versus Σ EQ #6 | EQ #9 versus Σ EQ #10 | EQ #11 versus Σ EQ #12 |
|--------------|-----|-----------------------------|------------------------------|-------------------------------|
| Record # | NEW | % Difference | % Difference | % Difference |
| NCCA-067 | 100 | 10.5% | 110.9% | 35.3% |
| NCCA-286 | 100 | -32.8% | 55.3% | -29.1% |
| NCCA-287 | 100 | -19.7% | 73.1% | -11.0% |
| NCCA-291 | 15 | -33.9% | -35.8% | -4.3% |
| NCCA-315 | 51 | 15.0% | 17.2% | 5.8% |
| NCCA-347 | 100 | -28.6% | 63.2% | -38.2% |
| NCCA-367 | 87 | -3.6% | 45.0% | 3.0% |
| NCCA-409 | 16 | -16.9% | 29.5% | -8.6% |
| NCCA-410 | 80 | -55.0% | -41.1% | 14.3% |
| NCCA-411 | 19 | -35.3% | -41.7% | -65.7% |
| NCCA-417 | 100 | -38.7% | 46.4% | -37.6% |
| NCCA-418 | 62 | -19.5% | 4.1% | -21.9% |
| NCCA-422 | 62 | -7.7% | 10.2% | -24.7% |
| NCCA-425 | 100 | -39.0% | 45.4% | -38.5% |
| NCCA-427 | 37 | -27.4% | -16.2% | -50.2% |
| NCCA-428 | 87 | 10.3% | 31.0% | 34.0% |
| NCCA-431 | 100 | -41.6% | 41.4% | -42.1% |
| NCCA-435 | 47 | -6.4% | -11.7% | -10.7% |
| NCCA-441 | 100 | -46.5% | 35.1% | -47.3% |
| NCCA-454 | 50 | -38.1% | -17.7% | -52.1% |
| NCCA-456 | 99 | -28.4% | 14.0% | -10.1% |
| NCCA-458 | 47 | -30.6% | -21.1% | -50.0% |
| MAD | | 26.6% | 36.7% | 28.8% |
| Predict (20) | | 40.9% | 31.8% | 36.4% |
| %Above | | 13.6% | 68.2% | 18.2% |
| %Below | | 86.4% | 31.8% | 81.8% |

Table 5-8: Summary of Program versus CSCI Differences

- Both of the CSCI-level traditional regressions tended to consistently estimate a smaller effort than the corresponding traditional program-level regressions. This effect is the exact opposite for the non-traditional regression. Upon inspection of the non-traditional equation (and the associated figures 5-5 through 5-8), it became apparent that the program-level regression discounts the nominal effort ($a * \text{Total SLOC}^b$) much more slowly (i.e., the program-level exponent is smaller, therefore the discount is smaller as the percent reuse increases) than the CSCI-level regression. However, the CSCI-level regression has a much larger constant. Therefore, because a majority of the programs are greater than 80 percent new, the constant vice the discount factor drives the resulting delta.
- Additionally, as Figure 5-6 demonstrates for the non-traditional equation, the critical reuse crossover value appears to be approximately 50 percent. More specifically, at 50 percent the smaller constant in the program-level non-traditional regression is overcome by the greater discount factor that is applied by the CSCI-level non-traditional regression. In fact, as the graph depicts, the delta between CSCI- and program-level data points generally increases as the percent reuse varies above or below the critical reuse value of 50 percent.
- The biggest differences between the program- and CSCI-level regressions occurred when estimating 100 percent new programs using the non-traditional equations. As shown in Appendix D, this is being driven by the non-traditional program-level equation, which estimates effort on average to be substantially less than the traditional equations, while the CSCI-level estimates remain stable across equations. This will not be a problem in practice, however, since NCCA recommends that CSCI-level regressions for 100 percent new programs not be used (see discussion on

Section 5 - Effort Analysis: Normalized Regressions

page 5-14). Except for 100 percent new programs estimated using the non-traditional equation, the regression differences were stable over the range of percent new.

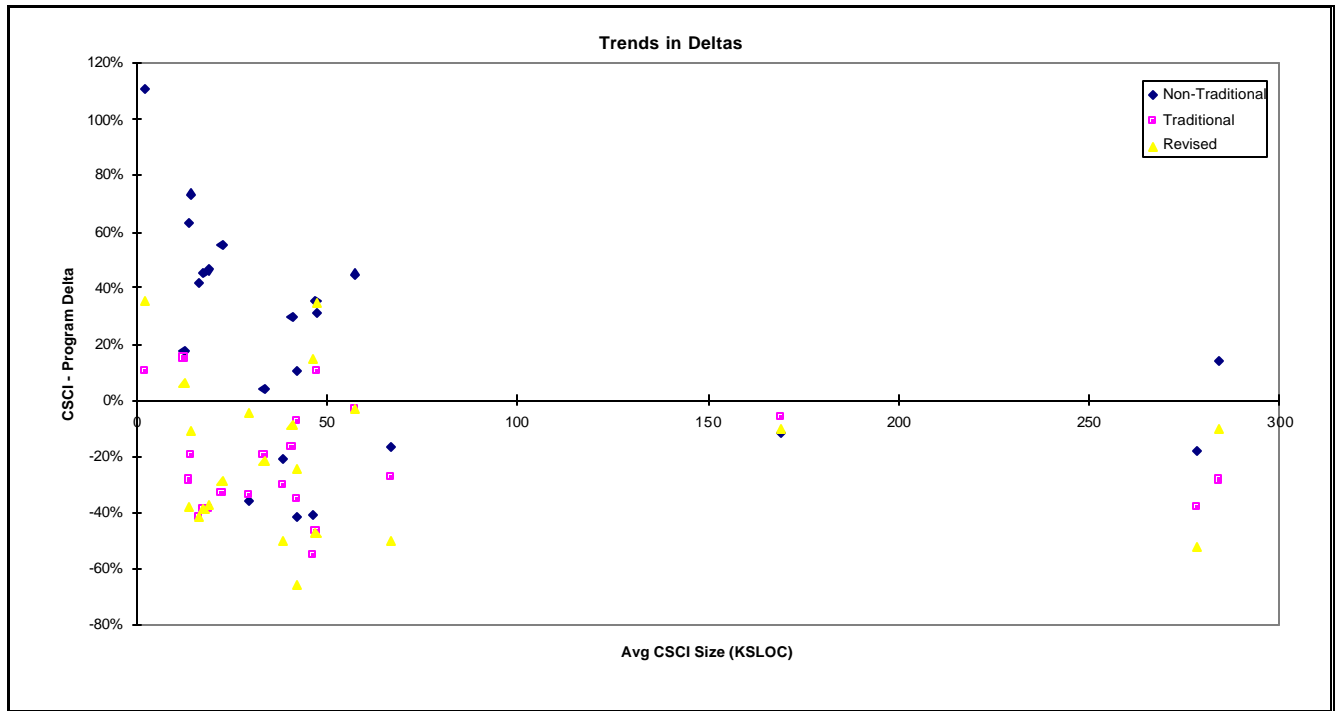


Figure 5-5: Average CSCI Size versus CSCI-Program Differences

Section 5 - Effort Analysis: Normalized Regressions

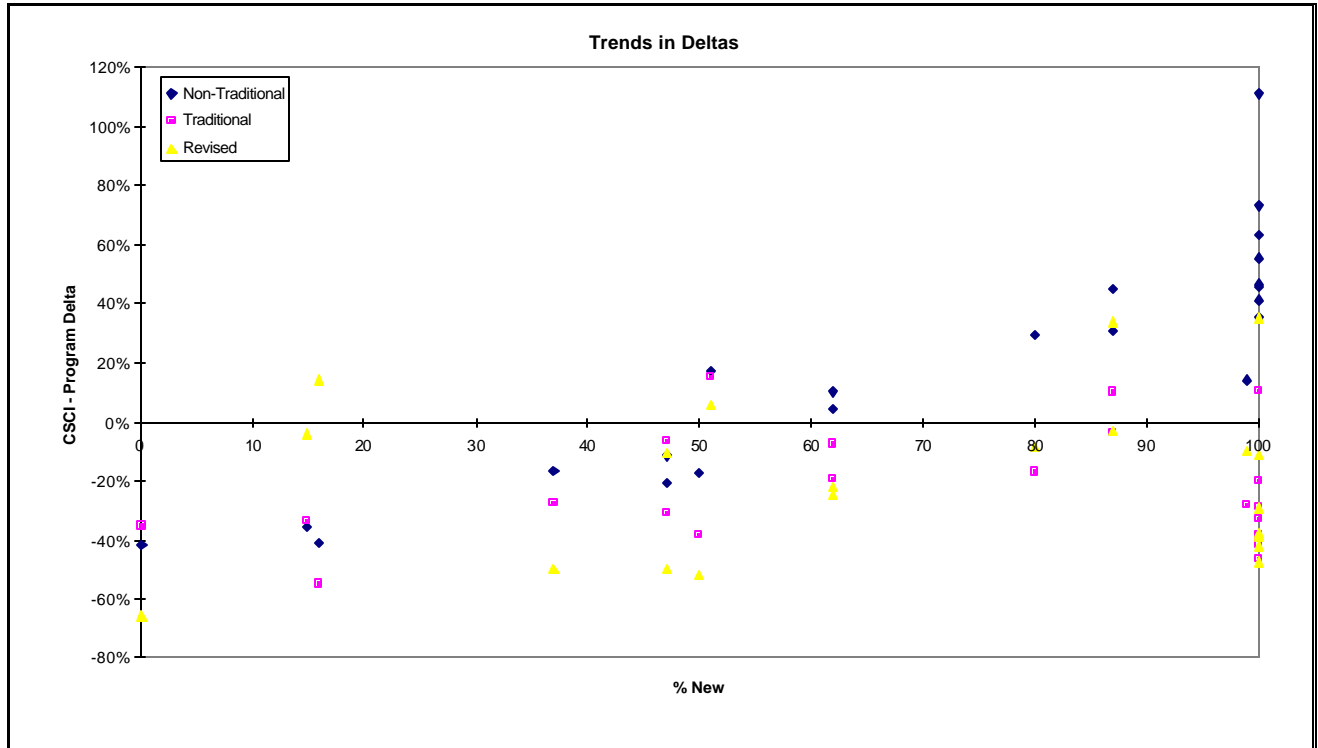


Figure 5-6: Percent New versus CSCI-Program Differences

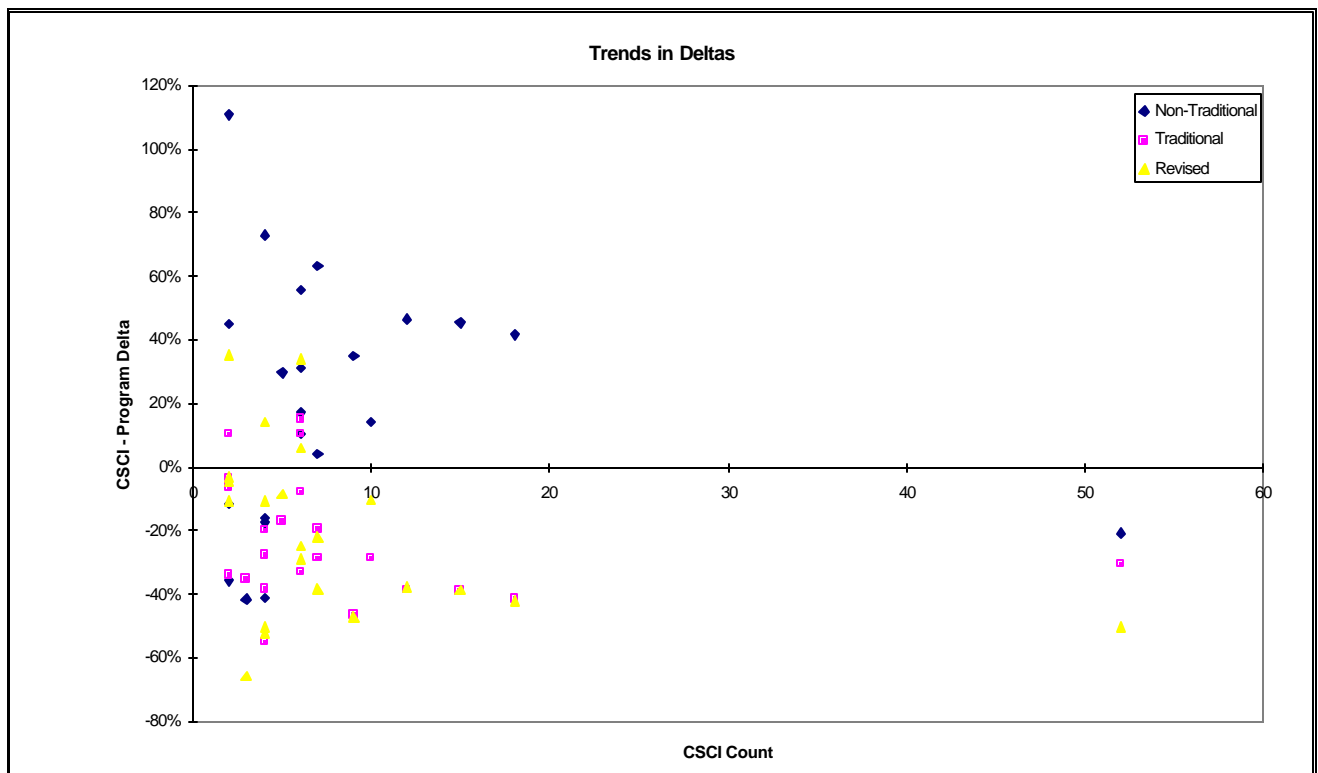


Figure 5-7: Program Size versus CSCI-Program Differences

Section 5 - Effort Analysis: Normalized Regressions

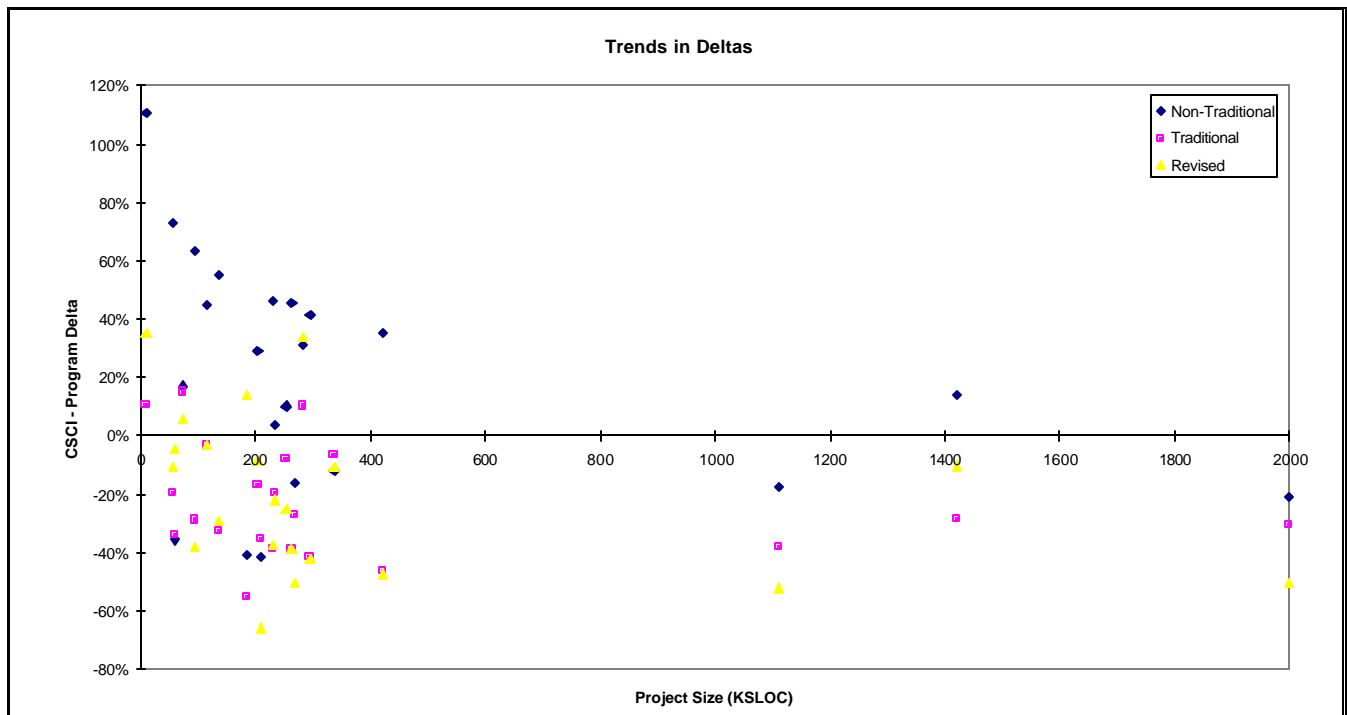


Figure 5-8: CSCI Count versus CSCI-Program Differences

- As the overall program size increased, the deltas between the CSCI- and program-level non-traditional regressions decreased, as shown in Figure 5-7.
- As expected, and demonstrated in the lower level spreadsheets provided in Appendix D, the revised traditional regression was consistently higher than the traditional top-level regression for program-level estimates of programs with large amounts of new code. Conversely, the revised traditional regression was consistently lower than the traditional regression for high-reuse program-level estimates, due to the discount factor. In contrast, the CSCI-level regressions tended to remain relatively stable across all code condition comparisons.

5.7 RECOMMENDATIONS

Table 5-9 shows the equation forms remaining for consideration at the program-level.

| Program-Level Equations | Effort = $\frac{1}{()}$ | Std Error | Predict (20) | Comments |
|---------------------------------|-------------------------------|-----------|--------------|---|
| Traditional - Top-Level | ESLOC; 100% New; Mode | 0.47 | 35% | Overestimates low-reuse data points & underestimates high-reuse data points |
| Non-Traditional | Total SLOC; (1-% reuse) | 0.52 | 18% | Flat discount rate; Does not account for all significant drivers (mode) |
| Revised Traditional - Top-Level | ESLOC; 100% New; Mode; %Reuse | 0.42 | 58% | Efactor = 1 |

Section 5 - Effort Analysis: Normalized Regressions

Table 5-9: Program-Level Equations Remaining

At the program-level, since the non-traditional equation appeared to discount programs too slowly and resulted in poorer statistics, the non-traditional equation was eliminated from further consideration. Hence, the top-level traditional and the revised traditional equation (which also accounted for high reuse programs) remained. The revised traditional equation improves the statistics of the regression and although the Efactor equals one, the 100% new and %Reuse dummies adjust for the differences in effort between new and reused code. Additionally, since the original traditional equation does not have dummy variables for both 100% new and high reuse programs, NCCA deleted it from any further consideration.

Table 5-10 presents the equation forms remaining for consideration at the CSCI-level.

| CSCI-Level Equations | Effort = $f(\quad)$ | Std Error | Predict (20) | Comments |
|---------------------------------|-------------------------------|-----------|--------------|---|
| Traditional - Top-Level | ESLOC; 100% New; Mode | 0.67 | 26% | Exponent<1 which implies economies of scale |
| Non-Traditional | SLOC; %New | 0.56 | 32% | Does not account for all significant drivers (mode) |
| Revised Traditional - Top-Level | ESLOC; 100% New; Mode; %Reuse | 0.66 | 29% | Counterintuitive; Efactor = 0 |

Table 5-10: CSCI-Level Equations Remaining

Because the revised traditional CSCI-level equation resulted in counterintuitive results (as percent reuse increased, effort also increased), it was eliminated from further consideration. The non-traditional equation has the best statistics, however, it does not account for development mode differences, which were proven to be a productivity driver in Section 4 - **Effort Analysis: Significant Drivers**. Since NCCA considers recognition of the significant drivers to be a crucial attribute, NCCA deleted the CSCI-level non-traditional equation from further consideration. The traditional equation does account for productivity drivers, however, since the exponent is less than one, it also implies economies of scale as previously discussed on page 5-14. As stated previously, NCCA expects that as the program size increases, the number of CSCIs will also increase. Therefore, the constant will be applied repeatedly, which will eventually negate the impact of the exponent (i.e., economies of scale will no longer be realized).

Based on the previously cited strengths and weaknesses and the statistics provided above, NCCA recommends using the revised traditional program-level, equation [5-11], and the traditional CSCI-level, equation [5-6], regressions as the standard NCCA effort tools. Based on the underlying databases (reused code compositions), these tools should be applied as follows:

- If the program being estimated is **100 percent new**, apply the following equation (revised traditional program-level equation with the 100 Percent new dummy variable enabled) at the program-level:

Revised Traditional Program-Level Equation

$$\text{Effort} = 0.0012 * (\text{New SLOC})^{[1.1979 + (0.0326 * D_1)]}$$

$R^2 = 0.96$ Std Error = 0.42 Predict (20) = 58% N = 31 Range = 9 - 1,113 EKSLOC

where D_1 equals one if the program is embedded and zero otherwise.

Section 5 - Effort Analysis: Normalized Regressions

- If the program being estimated has **greater than or equal to 82 percent reused code**, apply the following equation (revised traditional program-level equation with the percent reused dummy variable enabled) at the CSCI-level:

Revised Traditional Program-Level Equation

$$\text{Effort} = 0.0012 * [\text{New SLOC} + (1 * \text{Reused SLOC})]^{[1.0085 + (0.0326 * D_1)]}$$

$R^2 = 0.96$ $\text{Std Error} = 0.42$ $\text{Predict (20)} = 58\%$ $N = 31$ $\text{Range} = 9 - 1,113 \text{ EKSLOC}$
where D_1 equals one if the program is embedded and zero otherwise.

- If the program being estimated is neither 100 percent new nor greater than or equal to 82 percent reused code, and the reused code is **evenly distributed between modified and reused code**, apply the following equation (revised traditional program-level equation) at the program-level:

Revised Traditional Program-Level Equation

$$\text{Effort} = 0.0012 * [\text{New SLOC} + (1 * \text{Reused SLOC})]^{[1.1067 + (0.0326 * D_1)]}$$

$R^2 = 0.96$ $\text{Std Error} = 0.42$ $\text{Predict (20)} = 58\%$ $N = 31$ $\text{Range} = 9 - 1,113 \text{ EKSLOC}$
where D_1 equals one if the program is embedded and zero otherwise.

- If the program being estimated is neither 100 percent new nor greater than or equal to 82 percent reused code, and the reused code is **predominately verbatim**, apply the following equation (traditional CSCI-level equation) at the CSCI-level:

Traditional CSCI-Level Equation

$$\text{Effort} = 0.0229 * [\text{New SLOC} + (0.03 * \text{Reused SLOC})]^{[0.8609 + (0.0315 * D_1) + (0.0529 * D_2)]}$$

$R^2 = 0.77$ $\text{Std Error} = 0.67$ $\text{Predict (20)} = 26\%$ $N = 97$ $\text{Range} = 0.4 - 253.4 \text{ EKSLOC}$
where D_1 equals one if the program is 100% new and zero otherwise; and D_2 equals one if the program is embedded and zero otherwise

This overall approach for estimating programs with sufficient program definition is recommended for the following reasons:

- 1) Significant productivity drivers (100% new and mode), as defined in Section 4 - **Effort Analysis: Significant Drivers**, are accounted for in the equations.
- 2) The anticipated decrease in effort expected on high reuse programs is accounted for in the equations.
- 3) The differences expected in effort due to different reuse compositions (predominantly verbatim versus not predominantly verbatim) are accounted for in the equations.
- 4) None of the resulting equations consistently overestimates or underestimates effort (i.e., more balanced residuals than the lower-level regressions).
- 5) One hundred percent new programs, which have historically been more difficult to develop, are accounted for in the equations.

5.8 CONCLUSIONS

Over 300 regressions were performed on various subsets of the NCCA Raw Database, with different explanatory variables applied. Yet, of all these regressions, only a few had a standard error that was under 40 percent, and most of these could not be considered because they either had too few data points or resulted in counterintuitive results. Provided below is a list of the reasons why the resulting regressions experienced relatively large variances.

- 1) **Code Counting Definitions:** Even though the database classified the SLOC as physical or logical, there are many other ways that code can be counted. Some databases were careful to state that their SLOC count included only delivered SLOC, but others did not. At an even lower level, some programs only counted delivered executable SLOC, while others counted all delivered code. Unless the programs all come from the same developer, who counts the SLOC consistently, there will be variances in size from program to program due to different counting conventions.
- 2) **Definitions:** The NCCA Raw Database was normalized by the phasing of the reported effort (i.e., it included only data points which reported effort from SDR through FQT). However, NCCA could not, in all situations, normalize for the differences in labor categories included within these phases, or the actual taskings included. The NCCA Raw Database does not track effort by labor category or activity to show what is included in the total effort. However, NCCA did attempt to verify that all databases included “direct support” types of activities, such as Program Management, Quality Control and Documentation. This is another source of variance in the data and another reason why data collection and analogies should be contractor specific.
- 3) **Schedule:** The effort required to develop a program is dependent upon the schedule; a suboptimal schedule will increase development effort. There were not enough data points in the NCCA Raw Database to test the effect of schedule on the total effort. Even if there were, the database only tracks actual schedule. The initial staffing of a program is based on estimated schedule. If the initial estimate is too steep, then a program manager may have to add staff, which reduces overall productivity. Ultimately, the difference between the optimal schedule and the actual schedule may influence the results and account for differences experienced among programs.
- 4) **Non-Homogeneous Data:** The NCCA Raw Database includes both older and newer programs. Many of the programs come from the early to mid-1970s when punch cards, batch-mode processing, and slow computers resulted in larger compiler times.⁴⁶ The older programs are also characterized by:
 - Large amounts of Assembly,
 - Low amounts of reuse,
 - Less sophisticated tools, and
 - Storage and processing constraints.

⁴⁶ The Doty Model [16] actually had a factor to account for the amount of time it took for software to be compiled.

Section 5 - Effort Analysis: Normalized Regressions

Conversely, newer programs have:

- Low or no amount of Assembly,
- Higher amounts of reuse,
- Modern tools, and
- Faster processors with cheaper memory.

Intuitively, the older programs should have lower productivity than newer programs, but in actuality only minimal differences have been experienced. This is possibly because newer programs are not only more productive, but also more complex. The complexity of newer programs negates the productivity improvements made by implementing modern tools, faster processors, etc. Therefore, over time minimal differences are realized. Additionally, the NCCA Raw Database represents a combination of well-behaved and ill-behaved programs. It does not, however, contain any canceled programs. Some of the programs experienced code growth, effort growth, and schedule slippages, due to the following reasons:

- **Staffing:** The developer may have experienced an unexpectedly high rate of turnover on key personnel skills. Also, the developer may have experienced difficulty building up to a particular staffing level due to market constraints.
- **Requirements creep:** The customer may have demanded more functionality from the software. Additionally, energetic developers may have added unwanted features to make the product more desirable.
- **Reuse:** The code the developer wanted to reuse might have required software fixes or enhancements to meet the required functionality.

Databases, which tracked the initial estimates of size, schedule, and effort along with the resulting metrics, would also be good indicators of ill-behaved programs. However, the NCCA Raw Database does not track effort estimates, so NCCA cannot assess the behavior of the underlying programs. Similar to the issue of old versus new data points, the standard regressions tend to average out the effects of well-behaved and ill-behaved programs.

5) **Tools and Processes:** The process and tools used to develop software can play an important role. Unfortunately, many of the metrics that attempt to capture these factors are subjective. When available for a given data point, the NCCA Raw Database identifies the design methodology used to develop the software (waterfall, incremental, spiral and evolutionary). However, because this information was often not available, NCCA could not statistically test the significance of process. The database does not identify the type of tools used to develop the software. But, based on the fact that the database identifies development time, it is safe to assume that the database captures the effects of modern, as well as older, tools and practices.

In conclusion, collecting more data that is sensitive to the areas discussed above will lead to better overall top-level regressions. However, it is unlikely that the variance of top-level regressions will ever approach the lower variance of contractor-specific regressions. As such, NCCA's recommended approach to estimating software development effort is to gather

complete, detailed, contractor-specific data, and use this data to construct contractor-specific estimating relationships. Accordingly, the standard estimating relationships presented in this section should be used if, and only if, the analyst is unable to collect contractor-specific data relevant to the future software development effort being estimated.

5.9 FUTURE EFFORTS

Decreasing estimating variance and adapting regressions to be more sensitive to environmental variables rests solely on the ability to collect additional information in the future. The following are some areas that should be explored:

- 1) **Internal Reuse:** Programs that reuse portions of their code over and over have common SLOC. In the NCCA Raw Databases, NCCA mapped this type of SLOC into the verbatim SLOC category. However, a potential problem exists: a program may inaccurately state that it is 100 percent new, when in fact common SLOC will be reused within the same program. The common code can only be considered new once. Hence, the NCCA Raw Database should be updated to discriminate between SLOC reused from outside programs and organizations and SLOC that are reused in the same program.
- 2) **Schedule:** It is desirable to create an empirical model that includes both size and schedule. NCCA should also investigate how scheduling at the CSCI-level affects overall program productivity to determine whether there is an optimal sequence.
- 3) **WBS Activity:** If the NCCA Raw Database captured software development effort by standardized WBS activity, then activity-specific estimating relationships could be investigated. Perhaps programming is directly related to the size of the program, while program management support is tied more directly to the overall length of the schedule. Estimates based on these WBS activities would enable a more sensitive “what-if” analysis to be accomplished (especially in areas like acquisition reform or when changes in software development standards occur).
- 4) **Platform Integration:** The NCCA Raw Database has no information regarding the cost of integrating a group of software products into an aircraft's Operational Flight Program (OFP) or into a ship's combat system. These costs are typically not included in the OFP's or combat system's development cost, but rather are attributed to the host platform (i.e., the aircraft or ship).
- 5) **Non-Operational to Operational Ratios and Associated Effort:** A significant amount of operational code is usually required for software development, but to truly estimate the total effort required, data collection should focus on both operational and non-operational code.

EFFORT ANALYSIS: NON-NORMALIZED PRODUCTIVITY FACTORS

6.1 INTRODUCTION

NCCA performed an analysis of the NCCA Raw Database in order to provide analysts with productivity factors to estimate effort when the standard regressions (discussed in Section 5 - **Effort Analysis: Normalized Regressions**) are not appropriate. The standard regressions should not be utilized when either: 1) analogous or contractor specific data points exist; 2) the program being estimated does not meet NCCA's normalization criteria; or 3) sufficient definition to determine whether the program meets NCCA's normalization criteria is unavailable.

NCCA's normalization criteria are based on the analysis in Section 4 - **Effort Analysis: Significant Drivers** and are as follows:

Domain or Mission is weapon system,
Code counting convention is logical,
Code is written primarily in an HOL (more than 70 percent), and
Phasing is from SDR through FQT.

In addition, the following information must be defined for the program being estimated:

Code condition,
Software mode, and
Hours per man-month.

For situations where the regressions are not appropriate, NCCA has developed non-normalized top-level productivity factors. This section of the handbook is composed of the following four subsections:

- Data
- Methodology and Results
- Recommendations
- Conclusions

6.2 DATA

All data used in this analysis came from the NCCA Raw Database (Section 3 - **Software Database**). NCCA created 12 data sets from the NCCA Raw Database to develop the factors.

6.3 METHODOLOGY AND RESULTS

NCCA followed four steps to develop the productivity factors. First, NCCA selected the factors to be developed. Second, the specific data sets required for the analysis were filtered from the NCCA Raw Database. Third, the productivity was calculated for each data point. Fourth, the average productivity, standard deviation, and CV were calculated for each data set. The average productivity is the top-level productivity factor.

The factors NCCA developed were based on the results of the analysis in Section 4 -**Effort Analysis: Significant Drivers**. The factors were developed to address non-normalized programs, (i.e., programs that do not satisfy all of the criteria shown on the previous page). Six sets of top-level factors were developed. Each set contains two factors (i.e., a total of 12 factors were developed). One of the factors should be used when the code condition is unknown, the other should be used when the code condition is known. The factors for programs where the code condition is unknown are expressed in Hrs/Total SLOC. The factors for programs where the code condition is known are expressed in Hrs/ESLOC. The factors were developed in this manner to address code condition as a productivity driver.

The first set of factors was developed for situations where the normalized standard regressions cannot be utilized because the program being estimated is a MIS program. There were insufficient data points to create normalized MIS regressions. Hence, these are non-normalized factors. The second and third sets of factors were developed for situations where the normalized standard regressions cannot be utilized because the program being estimated is written in more than 30 percent Assembly (i.e., the HOL content is less than 70 percent). The fourth set of factors was developed for situations where the normalized standard regressions cannot be utilized because the program being estimated was sized using a physical code counting convention. The fifth set of factors was developed for situations where the normalized standard regressions cannot be utilized because the code counting convention used to size the program being estimated is unknown. The sixth set of factors was developed for situations where the program being estimated was sized using a logical code count (a normalization criterion), yet the standard regressions cannot be utilized because the phasing is unknown. The remainder of this section discusses how each of the 12 factors was developed.

6.3.1 MIS PROGRAMS

METHODOLOGY

Two data sets were filtered to develop the factors to be utilized for MIS program estimates. Since these are top-level non-normalized factors, phasing, mode, and hours per man-month were not used as criteria for these or any of the other factors.

1) Code condition is unknown (17 data points):

The data set used to develop this factor consisted of MIS programs written primarily in an HOL minus six outliers.⁴⁷ The following six data points were eliminated for the same reason they

⁴⁷ The data was not filtered on code condition. In this data set and those that follow, all data points where code condition was known were included in the code condition unknown data set, and treated as if the code condition was unknown.

Section 6 - Effort Analysis: Non-Normalized Productivity Factors

were eliminated from the “MIS” data set in Section 4 - **Effort Analysis: Significant Drivers** (see page 4-3): NCCA-39, NCCA-43, NCCA-55, NCCA-74, NCCA-92, and NCCA-157. NCCA concluded that the actual effort for each of these six CSCIs was not captured, but rather that the total effort for the program was allocated to the CSCI-level. Table 6-1 demonstrates how this data set was filtered from the NCCA Raw Database. The italicized letters are the field names in the NCCA Raw Database used for the filters. The non-italicized letters are the criteria utilized to filter the NCCA Raw Database.

2) Code condition is known (17 data points):

Because the code condition was known for all 17 data points (i.e., there were no blanks in the **New** field), this factor was developed from the same data set as the previous factor (see Table 6-1).

| Code Condition Unknown | Code Condition Known |
|--|--|
| <i>Mission</i> = MIS <i>Lang1</i> ≠ Assembly (Eliminated 6 outliers) | <i>Mission</i> = MIS <i>Lang1</i> ≠ Assembly (Eliminated 6 outliers) <i>New</i> ≠ blank |

Table 6-1: Productivity Factor (MIS Programs)

The productivity, expressed in Hrs/SLOC was then calculated for each data point in the MIS code condition **unknown** data set. A 152-hour per man-month rate was assumed for those programs that did not report actual rates.

The productivity, expressed in Hrs/ESLOC, was then calculated for each data point in the MIS code condition **known** data set. Again, a 152-hour per man-month rate was assumed for those programs that did not report actual rates.

The average productivity, standard deviation, CV, and resulting CV of the factor (CV_{est})⁴⁸ were then calculated for each data set. The average productivity of each data set constitutes the standard productivity factor.

RESULTS

The factors to be utilized when analogous or contractor specific data is not available, the program is **MIS**, and

1) Code condition is unknown = **0.6913 Hrs/Total SLOC**

CV = 86%; CV_{est} = 132%; n = 17

2) Code condition is known = **0.8240 Hrs/ESLOC**

Efactor = 0; CV = 72%; CV_{est} = 103%; n = 17

⁴⁸ NCCA performed an additional calculation to obtain a comparable CV of the estimate derived from the application of the standard factor. CV_{est} is the standard error of the estimate divided by the mean of the actual values:

$$CV_{est} = \frac{SEE_{dataset}}{\bar{Y}}$$

Section 6 - Effort Analysis: Non-Normalized Productivity Factors

Although not intentionally filtered in this manner, the data sets used to develop these factors consisted entirely of programs that were sized by a logical code count. Therefore, applying this factor to MIS programs sized by a physical code count may overestimate effort. Since these factors were developed from data sets consisting of programs written primarily in an HOL, they should not be applied to MIS programs written primarily in Assembly (although a MIS program written in Assembly is highly unlikely). Factors (1) and (2) are the only factors applicable to MIS, since all future factors will be developed from data sets consisting entirely of weapon systems.

6.3.2 WEAPON SYSTEM PROGRAMS - PRIMARILY ASSEMBLY

METHODOLOGY

Two data sets were filtered from the NCCA Raw Database to develop the factors for estimating programs written in more than 30 percent Assembly.

- 3) Code condition is unknown (68 data points):

The data set used to develop this factor consisted of weapon system programs written in more than 30 percent Assembly (i.e., less than 70 percent HOL). Table 6-2 demonstrates how this data set was actually filtered from the database.

- 4) Code condition is known (61 data points):

The data set used to develop this factor consisted of the 68 data points listed above minus seven data points for which the code condition was unknown (see Table 6-2).

| Code Condition Unknown | Code Condition Known |
|--|--|
| <i>HOL < 0.7</i> <i>Mission ≠ MIS or blank</i> ⁴⁹ | <i>HOL < 0.7</i> <i>Mission ≠ MIS or blank</i> <i>New ≠ blank</i> |

Table 6-2: Productivity Factor (30% Assembly Programs)

RESULTS

The factors to be utilized when additional analogous or contractor specific data is not available, the program is **written in more than 30 percent Assembly**, and

- 3) Code condition is unknown = **2.6504 Hrs/Total SLOC**

CV = 120%; CV_{est} = 177%; n = 68

- 4) Code condition is known = **3.0093 Hrs/ESLOC**

Efactor = 0.6; CV = 115%; CV_{est} = 168%; n = 61

These factors were developed from data sets that consisted of programs written, on average, in 82.4 percent Assembly. If the program being estimated consists of more than 82.4 percent

⁴⁹All programs which have a blank in the **Mission** field must also be eliminated since some of these programs may be MIS programs.

Section 6 - Effort Analysis: Non-Normalized Productivity Factors

Assembly, these factors may underestimate the effort. If the program being estimated consists of less than 82.4 percent Assembly, these factors may overestimate the effort. This and the next set of factors are the only ones applicable to programs written in Assembly, since all other factors were developed from data sets consisting entirely of programs written primarily in an HOL.

6.3.3 WEAPON SYSTEM PROGRAMS - 100 PERCENT ASSEMBLY

METHODOLOGY

Two data sets were filtered from the NCCA Raw Database to develop the factors for estimating programs written entirely in Assembly.

- 5) Code condition is unknown (40 data points):

The data set used to develop this factor consisted of weapon system programs written entirely in Assembly (i.e., zero percent HOL). Table 6-3 demonstrates how this data set was filtered from the database.

- 6) Code condition is known (38 data points):

The data set used to develop this factor consisted of the 40 data points listed above minus two data points for which the code condition was unknown (see Table 6-3).

| Code Condition Unknown | Code Condition Known |
|-------------------------------|-------------------------------|
| <i>HOL = 0</i> | <i>HOL = 0</i> |
| <i>Mission ≠ MIS or blank</i> | <i>Mission ≠ MIS or blank</i> |
| | <i>New ≠ blank</i> |

Table 6-3: Productivity Factor (100% Assembly Programs)

RESULTS

The factors to be utilized when analogous or contractor specific data is not available, the program is **written entirely in Assembly**, and

- 5) Code condition is unknown = **3.7383 Hrs/Total SLOC**

CV = 100%; CV_{est} = 132%; n = 40

- 6) Code condition is known = **3.9904 Hrs/ESLOC**

Efactor = 0.69; CV = 98%; CV_{est} = 125%; n = 38

Although it is highly unlikely that a MIS program would be written in Assembly, analysts must use caution if applying these factors to MIS programs since they were developed from data sets consisting entirely of weapon systems.

6.3.4 PHYSICAL CODE COUNTING CONVENTION

METHODOLOGY

Two data sets were filtered from the NCCA Raw Database to develop the factors for estimating programs sized with a physical code counting convention.

7) Code condition is unknown (18 data points):

The data set used to develop this factor consisted of weapon system programs that were written primarily in an HOL (greater than or equal to 70 percent), and sized according to a physical code counting convention. Table 6-4 demonstrates how this data set was filtered from the database.

8) Code condition is known (18 data points):

Because the code condition was known for all 18 data points, this factor was developed from the same data set as previously described (see Table 6-4).

| Code Condition Unknown | Code Condition Known |
|--|---|
| Count = P Mission ≠ MIS or blank HOL ≥ 0.7 | Count = P Mission ≠ MIS or blank HOL ≥ 0.7 New ≠ blank |

Table 6-4: Productivity Factor (Physical Programs)

RESULTS

The factors to be utilized when analogous or contractor specific data is not available, the **Code Counting Convention is physical**, and

7) Code condition is unknown = **0.6357 Hrs/Total SLOC**

CV = 124%; CV_{est} = 93%; n = 18

8) Code condition is known = **0.7350 Hrs/ESLOC**

Efactor = 0; CV = 104%; CV_{est} = 123%; n = 18

Although not intentionally filtered in this manner, the data sets used to develop these factors consisted entirely of embedded mode programs. Therefore, the analyst must use caution when applying these factors to non-embedded mode programs, since they may overestimate effort of non-embedded programs. To reiterate, these factors should not be applied to MIS programs or to programs written primarily in Assembly since they were developed from data sets consisting entirely of weapon system programs written primarily in an HOL.

6.3.5 UNKNOWN CODE COUNTING CONVENTION

METHODOLOGY

Two data sets were filtered from the NCCA Raw Database to develop the factors for estimating programs with an unknown code counting convention.

9) Code condition is unknown (273 data points):

The data set used to develop this factor consisted of weapon system programs that were written primarily in an HOL (greater than or equal to 70 percent). Table 6-5 demonstrates how this data set was filtered from the database.

10) Code condition is known (262 data points):

The data set used to develop this factor consisted of the 273 data points listed above minus 11 data points for which the code condition was unknown (see Table 6-5).

| Code Condition Unknown | Code Condition Known |
|---|--|
| <i>Mission ≠ MIS or blank HOL ≥ 0.7</i> | <i>Mission ≠ MIS or blank HOL ≥ 0.7 New ≠ blank⁵⁰</i> |

Table 6-5: Productivity Factor (Unknown Code Condition Programs)

RESULTS

The factors to be utilized when analogous or contractor specific data is not available, the **Code Counting Convention is unknown**, and

9) Code condition is unknown = **1.3238 Hrs/Total SLOC**

CV = 128%; CV_{est} = 196%; n = 273

10) Code condition is known = **1.6763 Hrs/ESLOC**

Efactor = 0.12; CV = 107%; CV_{est} = 215%; n = 262

Again, since these factors were developed from data sets consisting entirely of weapon system programs that were written primarily in an HOL, they should not be applied to programs written primarily in Assembly or to MIS programs.

6.3.6 LOGICAL CODE COUNTING CONVENTION - PHASING UNKNOWN

METHODOLOGY

Two data sets were filtered from the NCCA Raw Database to develop the factors for estimating programs with a logical code counting convention and phasing unknown.

⁵⁰The blanks in the **New field** must be eliminated manually. There is one program with zero percent new code. This data point should be included in the data set, although LOTUS will eliminate it along with the blanks in the field.

Section 6 - Effort Analysis: Non-Normalized Productivity Factors

11) Code condition is unknown (186 data points):

The data set used to develop this factor consisted of weapon system programs that were sized by a logical code counting convention and written primarily in an HOL (greater than or equal to 70 percent). Table 6-6 demonstrates how this data set was filtered from the database.

12) Code condition is known (185 data points):

The data set used to develop this factor consisted of the 186 data points listed above minus one data point for which the code condition was unknown (see Table 6-6).

| Code Condition Unknown | Code Condition Known |
|---|---|
| <i>COUNT = L</i> <i>Mission ≠ MIS or blank</i> <i>HOL ≥ 0.7</i> | <i>COUNT = L</i> <i>Mission ≠ MIS or blank</i> <i>HOL ≥ 0.7</i> <i>New ≠ blank</i> ⁵¹ |

Table 6-6: Productivity Factor (Logical Programs, Phasing Unknown)

RESULTS

The factors to be utilized when additional analogous or contractor specific data is not available, the **Code Counting Convention is logical, the phasing is unknown, and**

11) Code condition is unknown = **1.3360 Hrs/Total SLOC**

CV = 113%; CV_{est} = 182%; n = 186

12) Code condition is known = **1.8597 Hrs/ESLOC**

Efactor = 0.04; CV = 83%; CV_{est} = 161%; n = 185

These factors should not be applied to programs written in more than 30 percent Assembly or to MIS programs.

6.4 RECOMMENDATIONS

These top-level productivity factors should be applied **if, and only if**, contractor-specific data is unavailable and the NCCA standard regressions are not appropriate (i.e. the program being estimated does not meet NCCA's normalization criteria). These factors should only be applied as a last resort, rough order-of-magnitude estimate with the corresponding CVs clearly identified. If the program is not sufficiently defined (which is typically the case during the very early stages of its life cycle), then these factors may be applied for a quick, top-level estimate. However, analysts must realize, and state, that the factors are very top-level and the resulting estimates have large uncertainty bounds.

Table 6-7 provides a summary of the top-level non-normalized productivity factors. If the analyst is forced to utilize the top-level productivity factors, NCCA suggests that a thorough

⁵¹The blanks in the **New field** must be eliminated manually. There is one program with zero percent new code. This data point should be included in the data set, although LOTUS will eliminate it along with the blanks in the field.

Section 6 - Effort Analysis: Non-Normalized Productivity Factors

analysis of the data set underlying the factor be conducted to determine if the data set is truly analogous to the program being estimated.

| Application | Code Condition Unknown (Hrs/Total SLOC) | Code Condition Known (Hrs/ESLOC) |
|--|---|----------------------------------|
| MIS; Primary language is an HOL | 0.6913 | 0.8240 |
| Weapon System ; >30% Assembly | 2.6504 | 3.0093 |
| Weapon System; 100% Assembly | 3.7383 | 3.9904 |
| Weapon System; Physical Code Count; >70% HOL | 0.6357 | 0.7350 |
| Weapon System; Code Count Unknown; >70% HOL | 1.3238 | 1.6763 |
| Weapon System; Logical Code Count; Phasing Unknown; >70% HOL | 1.3360 | 1.8597 |

Table 6-7: Summary of Top-Level Productivity Factors

6.5 CONCLUSIONS

Since these factors were developed from a non-normalized data set, their weaknesses are much greater than their strengths. Analysts should make every effort possible to obtain contractor-specific data and to sufficiently define the program so that the use of these factors is truly a last resort. For example, if the size of the program being estimated is provided in physical SLOC, then historical programs which have collected metrics in physical SLOC should be obtained and used. Specific strengths and weaknesses of the top-level productivity factors are discussed below:

6.5.1 STRENGTHS

The strengths associated with the top-level factors are:

- 1) Language, mission and code condition (through the use of total SLOC and ESLOC) are recognized productivity drivers in every factor.
- 2) Separate factors exist for programs written primarily in Assembly and HOL and separate factors exist for MIS and weapon system programs.
- 3) The factors can be applied to both CSCI- and program-level data.

6.5.2 WEAKNESSES

The weaknesses associated with the top-level factors are:

- 1) Mode is a productivity driver, yet it is not addressed in any factor with the exception of the physical code counting convention factors (which were developed from data sets consisting entirely of embedded programs). The physical code counting convention factor may overestimate effort if applied to non-embedded mode programs. All of the remaining factors are based on a mixture of embedded and non-embedded mode programs. These factors may underestimate effort if applied to 100 percent embedded mode programs, and

Section 6 - Effort Analysis: Non-Normalized Productivity Factors

overestimate effort if applied to 100 percent non-embedded mode programs. However, programs are typically composed of both embedded and non-embedded CSCIs. If the factors must be utilized, NCCA suggests that the mode composition of the data set underlying the factor be thoroughly examined to determine if the mode of the program being estimated is analogous.

- 2) Phasing is a productivity driver, yet phase-specific data sets were not used to develop any of the factors. This means that the factors may underestimate or overestimate productivity, depending on how the phasing of the program being estimated compares to the phasing of the programs underlying the factor. If the top-level factors must be utilized, NCCA suggests that the phasing composition of the underlying data set used to develop the applicable factor be thoroughly examined to determine if the phasing of the program being estimated is analogous.
- 3) The factors were developed from data sets containing some programs where the hours per man-month rate was unknown. A 152-hours per man-month rate was assumed for these programs. This adds additional uncertainty to the factors, since the productivity of those programs was overestimated or underestimated if the actual hours per man-month utilized was less than or greater than 152. If the top-level factors must be utilized, NCCA suggests that the hours per man-month of the data set used to develop the applicable factor be thoroughly examined to determine if the hours per man-month rate for the program being estimated is analogous.
- 4) Factors are not contractor specific. As with the normalized standard NCCA regressions, NCCA recommends that analogous contractor-specific data be obtained to minimize variances.
- 5) Due to lack of data, there is no tool to estimate MIS programs sized by a physical code count, or MIS programs written entirely in Assembly (although this type of program is highly unlikely).
- 6) Use of these factors generates a significant amount of uncertainty since they are non-normalized and have large CVs.

As stated previously, these top-level productivity factors should be applied **if, and only if**, contractor specific data does not exist and the program being estimated does not satisfy NCCA's standard regression normalization criteria.

EFFORT ANALYSIS: OVERALL PROCESS

NCCA's recommended effort estimating process (illustrated in detail in Figure 7-1) consists of four major steps.

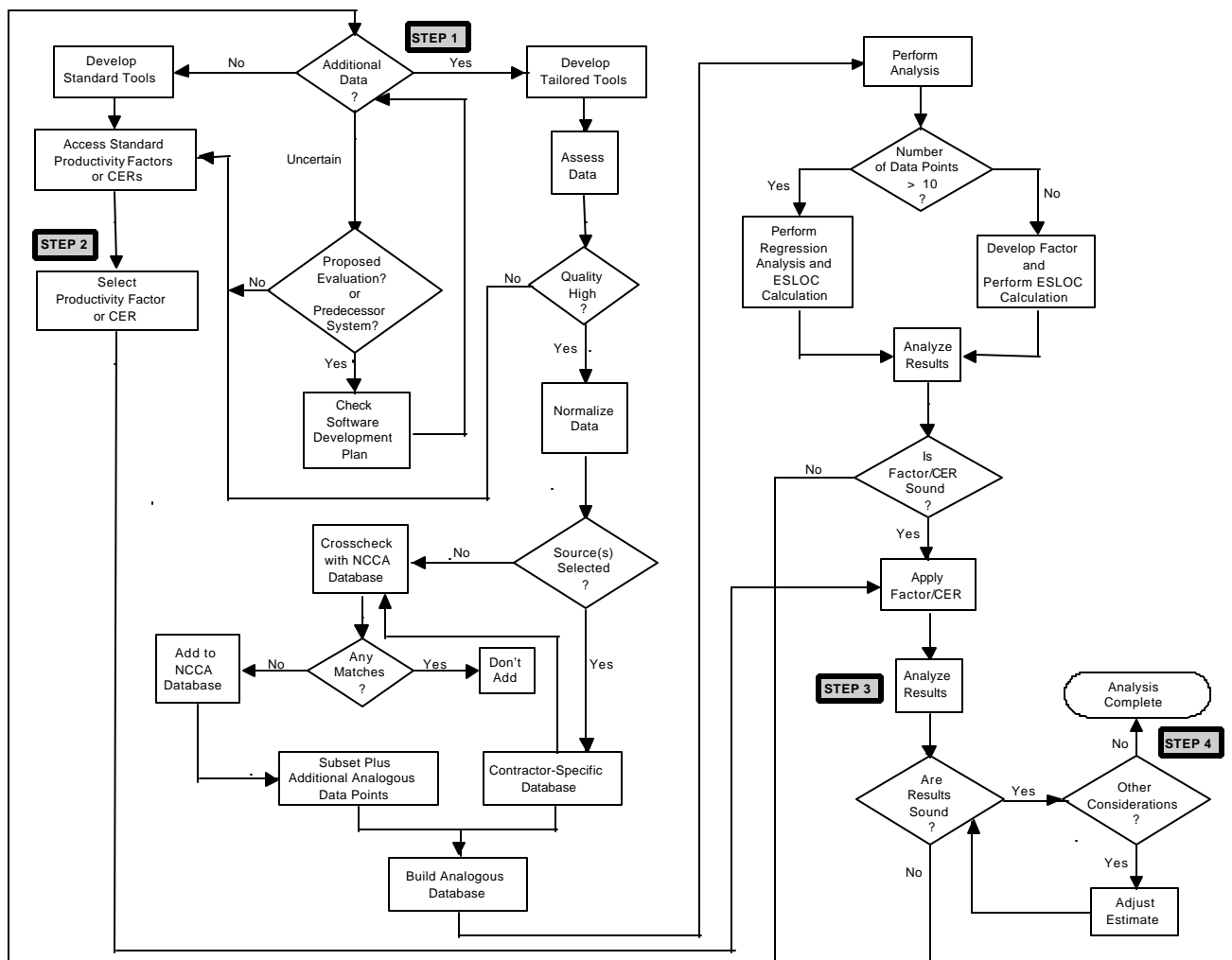


Figure 7-1: Effort Estimating Process

Step 1: Determine whether tailored or standard tools will be utilized.

- If contractor-specific data exists, assess the quality and applicability of the data, normalize

Section 7 - Effort Analysis: Overall Process

the data and develop tailored (contractor- or domain-specific) tools following the processes discussed in Sections 4-6.⁵²

- If contractor-specific data is unavailable, proceed to Step 2.

Step 2: Determine whether NCCA standard regressions or factors will be utilized.

If the program's

Domain is weapon system,
Counting convention is logical,
Development language is an HOL,
Development phases span SDR through FQT,
Code condition is known,
Development mode is known, and
Hours per man-month rate is known,

then the program satisfies NCCA's normalization criteria and NCCA standard regressions should be used as follows:

- If the program being estimated is **100 percent new**, apply the following equation (revised traditional program-level equation with the 100 percent new dummy variable enabled) at the program-level:

Revised Traditional Program-Level Equation

| |
|---|
| $\text{Effort} = 0.0012 * (\text{New SLOC})^{[1.1979 + (0.0326 * D_1)]}$ <p>R² = 0.96 Std Error = 0.42 Predict (20) = 58% N = 31 Range = 9 - 1,113 EKSLOC where D₁ equals one if the program is embedded and zero otherwise.</p> |
|---|

- If the program being estimated has **greater than or equal to 82 percent reused code**, apply the following equation (revised traditional program-level equation with the percent reused variable enabled) at the program-level:

Revised Traditional Program-Level Equation

| |
|--|
| $\text{Effort} = 0.0012 * [\text{New SLOC} + (1 * \text{Reused SLOC})]^{[1.0085 + (0.0326 * D_1)]}$ <p>R² = 0.96 Std Error = 0.42 Predict (20) = 58% N = 31 Range = 9 - 1,113 EKSLOC where D₁ equals one if the program is embedded and zero otherwise.</p> |
|--|

- If the program being estimated is neither 100 percent new nor greater than or equal to 82 percent reused code, and the reused code is **evenly distributed between modified and verbatim code**, apply the following equation (revised traditional program-level equation) at the program-level:

⁵²As depicted in Figure 5-1, if contractor-specific data is available, the analyst should develop "tailored" regressions, vice utilizing the NCCA standard regressions. This data should also be added to the NCCA Database using the procedures referenced in Chapter 2 - **Defining the Problem** and provided in Appendix A, NCCA Historical Software Data Request Form's Mapping Procedures.

Section 7 - Effort Analysis: Overall Process

Revised Traditional Program-Level Equation

$$\text{Effort} = 0.0012 * [\text{New SLOC} + (1 * \text{Reused SLOC})]^{[1.1067 + (0.0326 * D_1)]}$$

$R^2 = 0.96$ $\text{Std Error} = 0.42$ $\text{Predict (20)} = 58\%$ $N = 31$ $\text{Range} = 9 - 1,113 \text{ EKSLOC}$
where D_1 equals one if the program is embedded and zero otherwise.

- If the program being estimated is neither 100 percent new nor greater than or equal to 82 percent reused code, and the reused code is **predominantly verbatim**, apply the following equation (traditional CSCI-level equation) at the CSCI-level:

Traditional CSCI-Level Equation

$$\text{Effort} = 0.023 * [\text{New SLOC} + (0.03 * \text{Reused SLOC})]^{[0.8609 + (0.0315 * D_1) + (0.0529 * D_2)]}$$

$R^2 = 0.77$ $\text{Std Error} = 0.67$ $\text{Predict (20)} = 26\%$ $N = 97$ $\text{Range} = 0.4 - 253.4 \text{ EKSLOC}$
where D_1 equals one if the program is 100% new and zero otherwise; and D_2 equals one if the program is embedded and zero otherwise

If the program does not satisfy NCCA's normalization criteria, then NCCA standard productivity factors should be utilized as follows:

If the program is MIS, and

- 1) code condition is unknown, productivity = 0.6913 hrs/Total SLOC
- 2) code condition is known, productivity = 0.8240 hrs/ESLOC; Efactor = 0

If the program is written significantly (>30%) in Assembly, and

- 1) code condition is unknown, productivity = 2.6504 hrs/Total SLOC
- 2) code condition is known, productivity = 3.0093 hrs/ESLOC; Efactor = 0.6

If the program is written entirely in Assembly, and

- 1) code condition is unknown, productivity = 3.7383 hrs/Total SLOC
- 2) code condition is known, productivity = 3.9904 hrs/ESLOC; Efactor = 0.69

If the counting convention is physical, and

- 1) code condition is unknown, productivity = 0.6357 hrs/Total SLOC
- 2) code condition is known, productivity = 0.7350 hrs/ESLOC; Efactor = 0

If the counting convention is unknown, and

- 1) code condition is unknown, productivity = 1.3238 hrs/Total SLOC
- 2) code condition is known, productivity = 1.6763 hrs/ESLOC; Efactor = 0.12

If the counting convention is logical, but phasing is unknown, and

- 1) code condition is unknown, productivity = 1.3360 hrs/Total SLOC
- 2) code condition is known, productivity = 1.8597 hrs/ESLOC; Efactor = 0.04

Step 3: Analyze results for reasonableness.

Step 4: Consider other conditions that may affect productivity (such as those discussed in the NCCA issue papers).

As discussed above, NCCA's recommended effort estimating process requires the analyst to apply standard regressions or factors if contractor specific data is not available. Table 7-1 compares the performance of these standard regressions and factors. The same database

Section 7 - Effort Analysis: Overall Process

used in Section 5 - **Effort Analysis: Normalized Regressions** is displayed; however, in this case, the appropriate cost estimating tool is applied (i.e., if the program was written in Assembly, then the Assembly factor was utilized). The regressions clearly outperform the productivity factors (Predict (20) equals 88 percent for regressions versus seven percent for factors, and the corresponding MAD equals 13 percent for regressions versus 73 percent for factors). **Hence, the factors should only be used as a last resort in any cost estimating scenario.**

| Overall Software Development Cost Estimating Comparison | | | | | | | | | | |
|---|-----------|-----------|-----------|---------|---|----------|----------|---------|-------------------------------------|---|
| PROGRAM | TOTAL | NEW | NOT NEW | MOD | | NCCA | ACTUAL | | | |
| NAME | LOC | LOC | LOC | LOC | METHODOLOGY USED | ESTIMATE | EFFORT | DELTA | | NOTES |
| NCCA-067 | 11,036 | 11,036 | 0 | 0 | Physical Count Prod Factor | 53 | 27 | 95.90% | | From Navy Internal Data, Count was physical |
| NCCA-286 | 134,700 | 134,700 | 0 | 0 | Top level Prod Factor | 1,486 | 2,563 | -42.00% | | From Silver DB, Count, hrs/mm.scope of effort unknown |
| NCCA-287 | 57,000 | 57,000 | 0 | 0 | Top level Prod Factor | 629 | 1,720 | -63.50% | | From Silver DB, Count, hrs/mm.scope of effort unknown |
| NCCA-291 | 58,504 | 8,893 | 49,611 | 2,223 | Top Level CSCI CER w 100% new & Emb dummies | 76 | 73 | 4.80% | | From SEL, Included in normalized DB |
| NCCA-315 | 74,770 | 37,759 | 37,011 | 8,075 | Logical Count, Unknown Phases Prod. Factor | 480 | 164 | 192.70% | | Actual effort does not include requirements |
| NCCA-347 | 95,120 | 95,120 | 0 | 0 | Logical Count, Unknown Phases Prod. Factor | 1,164 | 642 | 81.30% | | Actual effort does not include requirements |
| NCCA-367 | 114,361 | 99,952 | 14,409 | 10,407 | Top Level Program (Revised) | 493 | 497 | -0.70% | | From SEL, Included in normalized DB |
| NCCA-409 | 204,000 | 163,200 | 40,800 | 15,503 | Top Level CSCI CER w 100% new & Emb dummies | 1,326 | 1,231 | 7.70% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-410 | 185,600 | 30,253 | 155,347 | 155,347 | Top Level Program (Revised) | 380 | 335 | 13.50% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-411 | 210,000 | 40,000 | 170,000 | 20,000 | Physical Count Prod Factor | 193 | 164 | 25.90% | | From Navy Internal Data, Count was physical |
| NCCA-417 | 231,870 | 231,870 | 0 | 0 | Top Level Program (Revised) | 4,979 | 5,103 | -2.40% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-418 | 234,130 | 144,458 | 89,672 | 0 | Top Level CSCI CER w 100% new & Emb dummies | 1,252 | 2,350 | -46.70% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-422 | 254,142 | 158,036 | 96,106 | 96,106 | Logical Count, Unknown Phases Prod. Factor | 1,981 | 4,421 | -55.20% | | From SMC DB, Evolutionary Program |
| NCCA-425 | 263,179 | 263,179 | 0 | 0 | Top level Prod Factor | 2,902 | 5,244 | -44.70% | | From Silver DB, From Silver DB, From Silver DB, |
| NCCA-427 | 267,900 | 98,587 | 169,313 | 0 | Top level Prod Factor | 1,311 | 864 | 51.90% | | From Silver DB, Count, hrs/mm.scope of effort unknown |
| NCCA-428 | 283,500 | 245,511 | 37,989 | 0 | Top level Prod Factor | 2,758 | 1,944 | 41.80% | | From Silver DB, Count, hrs/mm.scope of effort unknown |
| NCCA-431 | 334,704 | 334,704 | 0 | 0 | >30% Assembly Prod Factor | 6,626 | 7,592 | -12.70% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-435 | 338,088 | 157,887 | 180,201 | 180,201 | >30% Assembly Prod Factor | 5,266 | 1,850 | 184.70% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-441 | 422,552 | 422,552 | 0 | 0 | Logical Count, Unknown Phases Prod. Factor | 5,170 | 2,920 | 77.10% | | Program was concept exploration |
| NCCA-454 | 1,113,000 | 556,500 | 556,500 | 556,500 | Top Level Program (Revised) | 9,637 | 10,976 | -12.20% | | Mitre Non-Ada DB, Included in normalized DB |
| NCCA-456 | 1,420,872 | 1,406,663 | 14,209 | 14,209 | Physical Count Prod Factor | 6,802 | 4,652 | 46.20% | | From Navy Internal Data, Count was physical |
| NCCA-458 | 1,997,934 | 933,575 | 1,064,359 | 0 | Top Level CSCI CER w 100% new & Emb dummies | 9,587 | 8,193 | 17.00% | | From Navy Internal, Included in normalized DB |
| | | | | | | | MAD | 50.90% | General Methodology | |
| | | | | | | | P(20) | 36.40% | | |
| | | | | | | | %Above 0 | 59.10% | | |
| | | | | | | | %Below 0 | 40.90% | | |
| | | | | | | | P(20) | 87.50% | CER portion of Methodology | |
| | | | | | | | MAD | 13.10% | | |
| | | | | | | | %Above 0 | 62.50% | | |
| | | | | | | | %Below 0 | 37.50% | | |
| | | | | | | | P(20) | 7.10% | Productivity Portion of Methodology | |
| | | | | | | | MAD | 72.50% | | |
| | | | | | | | %Above 0 | 71.40% | | |
| | | | | | | | %Below 0 | 28.60% | | |

Table 7-1: Regression versus Factor Performance

SCHEDULE ANALYSIS

8.1 INTRODUCTION

The purpose of this section is to describe the methodology and procedures used to develop a software schedule estimating tool. In recent years, the importance of the software development schedule has increased because it is often on the critical path in weapon system developments. However, software schedule estimates tend to be optimistic, with programs typically experiencing between 20 and 60 percent schedule growth. Typical reasons for schedule growth include changes in requirements, unrealistic project planning, and staffing problems. Regardless of the reason for the delay or slip in schedule, it will ultimately result in cost growth.

NCCA has developed both top-level factors and schedule estimating relationships that estimate schedule in months. The methodology and recommended approaches are addressed below. However, these recommended approaches should only be used when additional analogous or contractor specific data are not available. The following subsections will describe the schedule analysis:

- NCCA Schedule Databases
- Methodology and Results
- Recommendations
- Conclusions
- Future Efforts

8.2 NCCA SCHEDULE DATABASES

Similar to the effort databases, NCCA created a separate schedule database to support schedule analyses. The detailed methodology will be provided below.

8.2.1 GROUND RULES AND ASSUMPTIONS

The general ground rules and assumptions followed to create the NCCA Schedule Database are:

- 1) Only program-level data points were used for the NCCA Schedule Database.
- 2) The schedule dates were assumed to be the midpoint of the month.
- 3) All SLOC are logical code.
- 4) All programming languages, except Assembly, were defined as HOL.

8.2.2 RAW SCHEDULE DATABASE

The NCCA Raw Schedule Database is a subset of the NCCA Raw Database detailed in Section 3 - **Software Database**. A record in the NCCA Raw Database contains 73 data fields; however only 48 of the 73 data fields were used in the NCCA Raw Schedule Database. Of the 48 data fields, the following fields (see Section 3 – Software Database for definitions) were predominantly used:

- 1) Program name
- 2) Platform
- 3) Program- or CSCI-level
- 4) Size (total SLOC)
- 5) Programming language
- 6) Effort expended (man-months)
- 7) Duration (total schedule in calendar months)
- 8) Acquisition period
- 9) Code counting convention
- 10) Software development phase

A query of the NCCA Raw Database (references [5] through [7]) was performed and a total of 151 program-level data points, which included schedule as well as effort, were obtained. These 151 data points were screened to identify those having schedule dates from SDR through FQT. Thirty-seven of the 151 points met this criterion and were retained. These data points constituted the NCCA Raw Schedule Database.

The 37 data points were a mixture of HOL and Assembly language programs. The mission types included C³, radar, missile, ASW, and simulation programs installed on air, ship, and ground platforms. The total SLOC ranged from 2.3 to 1,113 KSLOC and total development effort ranged from 11 to 10,976 man-months. The associated schedule ranged from 5 to 74 months, with a mean schedule of 27.9 months.

See Appendix E for a description of the data points in the NCCA Raw Schedule Database.

8.2.3 NCCA NORMALIZED SCHEDULE DATABASE

The 37 data points in the NCCA Raw Schedule Database were then screened to obtain the NCCA Normalized Schedule Database. The criteria for the NCCA Normalized Schedule Database were: 1) the effort occurred from SDR through FQT and 2) the hours per man-month were known and converted to 152 hours per man-month. One data point was eliminated because the hours per man-month rate was unknown. Four additional points were deleted because the effort included the SIT phase or the OTE phase, which are outside the SDR through FQT scope of effort. Finally, 16 additional data points were deleted because the Software Requirements Analysis Phase (SDR-SSR) was not included in the effort, leaving 16 normalized data points (i.e., effort and schedule from SDR through FQT).

The 16 normalized data points, see Appendix E, were a mixture of HOLs (e.g., Fortran, Jovial, CMS-2) and Assembly language programs. However, none of the programs were written in Ada. The mission types are characterized as C³, radar, and simulation programs, which were

Section 8 - Schedule Analysis

installed on air, ship, and ground platforms. The total SLOC ranged from 20 to 1,113 KSLOC, and total development effort ranged from 157 to 10,976 man-months. The associated schedule ranged from 12 to 74 months. The mean schedule was 33 months.

Figure 8-1 shows the filtering process that resulted in the NCCA Raw Schedule and NCCA Normalized Schedule Databases used in developing the software schedule regressions.

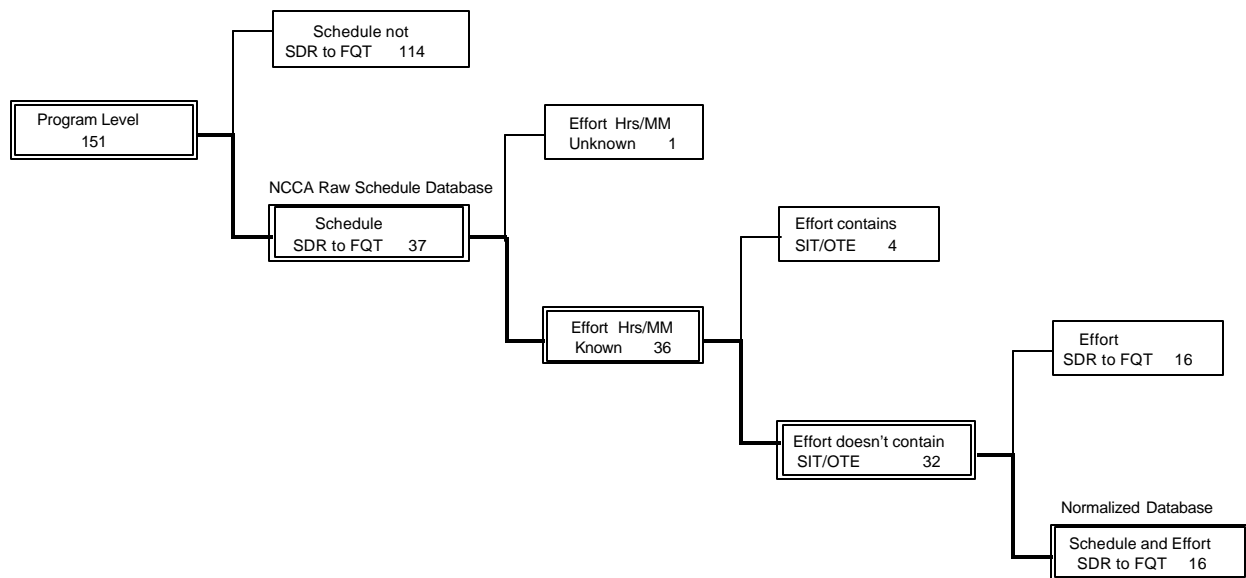


Figure 8-1: NCCA Schedule Databases

Since the NCCA Normalized Schedule Database was small ($N = 16$), the Mann-Whitney U test, a non-parametric test, was performed to determine if the non-normalized and normalized data points could be combined. The Wilcoxon Two-Sample test was performed to determine whether the data points within the NCCA Normalized Schedule Database should be separated due to language or mode differences. Essentially, these non-parametric tests compared and tested the means of both samples to determine if they were from the same population and could, be combined into one database. Additional information about these non-parametric tests is available in Appendix C and references [14] and [19].

The results of the Mann-Whitney U Test indicated that non-normalized and normalized data points should not be combined. Results of this test are found in Appendix E.

The Wilcoxon Two-Sample test was performed on the NCCA Normalized Schedule Database ($N = 16$). The results indicated that the HOL and Assembly language data points should remain combined. Results of this test are found in Appendix E.

The Wilcoxon Two-Sample test was also performed on the NCCA Normalized Schedule Database to determine if the two semi-detached programs and 14 embedded programs were from the same population. Even though there were so few data points, the results indicated that the semi-detached and embedded programs could remain combined. Results of this test are found in Appendix E.

Section 8 - Schedule Analysis

Based on the non-parametric test results summarized above, the final NCCA Normalized Schedule Database consisted of all 16 data points, with HOL and Assembly languages and semi-detached and embedded modes combined. This database was used to develop the factors and schedule estimating relationships. The NCCA Normalized Schedule Database is shown in Table 8-1.

| Record | Platform Type | Mission Area | Lang1 | Total SLOC | Total Man-Months | Total Months |
|----------|---------------|--------------|----------|------------|------------------|--------------|
| NCCA-146 | | Radar | CMS-2 | 20,276 | 241 | 32 |
| NCCA-183 | Ground | C3 | EDL | 26,200 | 289 | 28 |
| NCCA-284 | | C3 | Fortran | 56,021 | 157 | 17 |
| NCCA-298 | Ground | C3 | Assembly | 63,944 | 469 | 12 |
| NCCA-360 | | C3 | Fortran | 102,806 | 1222 | 32 |
| NCCA-390 | | C3 | Fortran | 139,527 | 586 | 25 |
| NCCA-404 | Ground | SIM | ? | 169,000 | 704 | 21 |
| NCCA-409 | Ship | Radar | Jovial | 185,600 | 335 | 25 |
| NCCA-410 | Ship | Radar | Jovial | 204,000 | 1231 | 32 |
| NCCA-417 | Ground | C3 | CMS-2 | 231,870 | 5103 | 74 |
| NCCA-418 | | Radar | Jovial | 234,130 | 2350 | 39 |
| NCCA-426 | Air | C3 | Fortran | 263,992 | 6496 | 42 |
| NCCA-429 | Air | C3 | Assembly | 285,400 | 1326 | 22 |
| NCCA-431 | | C3 | Assembly | 295,196 | 7592 | 70 |
| NCCA-435 | | C3 | Fortran | 338,088 | 1850 | 21 |
| NCCA-454 | | C3 | Jovial | 1,113,000 | 10976 | 40 |

Table 8-1: NCCA Normalized Schedule Database

The primary strength of the NCCA Normalized Schedule Database is that it includes only programs that had effort and schedule from SDR through FQT. Another strength is that the schedule range is robust; it includes programs ranging from one to six years, 20 to 1,113 KSLOC and various mission types, including both C³ and radar systems. The weaknesses include: 1) 14 of the 16 data points' are embedded mode; 2) there are no Ada data points; 3) the data points represent older programs (late 1970s through mid-1980s); and 4) the data points are extracted from only one source database (MITRE Non-Ada Database).

8.3 METHODOLOGY AND RESULTS

Using the NCCA Normalized Schedule Database, two analytical approaches (factors and equations) were investigated.

8.3.1 APPROACH ONE

To develop factors, the mean and median statistics for schedule in total calendar months, effort in total MM, and size in KSLOC were calculated. Appendix E contains the non-parametric tests performed on these data sets. Table 8-2 shows the results. The means and medians were then utilized to partition the database (e.g., programs less than or equal to the median KSLOC versus programs greater than the median KSLOC). As shown in Table 8-3, the mean schedule in months, the Predict (20) and the associated CV_{est} were calculated for each partition. (See Appendix C for a discussion of the CV_{est} calculation and Appendix E for supporting spreadsheets.) The recommended schedule estimating approach was selected based on the partition that minimized the CV or CV_{est}.

Section 8 - Schedule Analysis

| | |
|----------------------------------|--------------------|
| Months Range = 12 to 74 | Mean = 33 Months |
| | Median = 30 Months |
| KSLOC Range = 20 to 1,113 | Mean = 233 KSLOC |
| | Median = 195 KSLOC |
| Man-Months Range = 157 to 10,976 | Mean = 2558 MM |
| | Median = 1227 MM |

Table 8-2: NCCA Normalized Schedule Database Partitions

| | Partitions | N | Mean (Months) | CV _{est} | PREDICT (20) |
|---|-------------------------------------|----|---------------|-------------------|--------------|
| 1 | All Programs | 16 | 33 | 0.54 | 0.38 |
| 2 | Programs ≤ 233 KSLOC (Mean KSLOC) | 10 | 30 | 0.52 | 0.56 |
| | Programs > 233 KSLOC (Mean KSLOC) | 6 | 39 | | |
| 3 | Programs ≤ 195 KSLOC (Median KSLOC) | 8 | 24 | 0.45 | 0.44 |
| | Programs > 195 KSLOC (Median KSLOC) | 8 | 43 | | |
| 4 | Programs ≤ 1227 MM (Median MM) | 8 | 24 | 0.45 | 0.44 |
| | Programs > 1227 MM (Median MM) | 8 | 43 | | |
| 5 | Programs ≤ 2558 MM (Mean MM) | 12 | 26 | 0.32 | 0.50 |
| | Programs > 2558 MM (Mean MM) | 4 | 57 | | |

Table 8-3: Statistical Results of Partitions

Partition 5 has the smallest resulting CV_{est} and the following results:

- 1) For programs with Estimated Effort ≤ 2558 MM: Schedule = 26 months
 - 2) For programs with Estimated Effort > 2558 MM: Schedule = 57 months
- CV_{est} = 32%, n = 16, Predict (20) = 50%

The resulting Predict (20) indicates that 50 percent of the schedules estimated with these factors were within 20 percent of the actual schedules.

The primary strength of the factors is the associated statistics. The residuals of the factors appeared to show no bias, and the approach is simple and easy to use. Also the factors broadly distinguish between large and small programs; however, the lack of sensitivity within the broad groupings is a primary weakness of this approach (i.e., the factors estimate the same schedule for a program with effort equal to 2559 man-months as they do for a program with effort equal to 4000 man-months). Finally, the factors may not be good estimators for Ada data points because the database has no Ada data points. However, NCCA has not found any documented evidence that suggests language impacts the software development schedule.

8.3.2 APPROACH TWO

In addition to factors, regressions were investigated. Linear as well as exponential and power equations were developed with the exponential form showing the most promise. In all of the regressions, the dependent variable was actual schedule expressed in calendar months. The independent variables included total effort in man-months, and total SLOC, ESLOC, or productivity (man-months/ESLOC). Based on the results of the analysis in Section 4 - **Effort Analysis: Significant Drivers**, dummy variables included mode (i.e., embedded versus semi-detached) and code condition (i.e., 100 percent new programs versus not 100 percent new

programs). Additionally, NCCA developed regressions, which included various percent reused, and KSLOC trade-offs.

NCCA analyzed the regressions and determined the most promising ones. Only significant regressions (at the 95 percent confidence level) were considered. The list of non-significant equations and their associated statistics is shown in Appendix E. The following six significant normalized equations remained. They are divided into the following categories: 1) traditional equations, where software schedule is a function of estimated effort in man-months and 2) non-traditional equations, where schedule is a function of estimated size in ESLOC. Similar to the effort analysis (Sections 4 through 7), NCCA quantitatively solved for the Efactor which converted reused code to equivalent new code, while minimizing the standard error. Appendix E contains the regression plots and residual analyses for equations [8-1] through [8-6].

Traditional Equations: Actual Schedule = f(Estimated Effort)

$$\text{Schedule (Months)} = 4.87 * (\text{MM})^{0.2556}$$

$R^2 = 0.50$; CV = 0.35; Predict (20) = 44%; Range = 241 - 10,976 MM [8-1]

$$\text{Schedule (Months)} = 5.12 * (\text{MM})^{0.2266} * e^{(0.3574 * D_1)}$$

$R^2 = 0.64$; CV = 0.31; Predict (20) = 44%; Range = 241 - 10,976 MM [8-2]
where dummy variable $D_1 = 1$ for programs 100% New and 0 otherwise

$$\text{Schedule (Months)} = 6.50 * (\text{MM})^{0.2320} * e^{(-0.3883 * D_1)}$$

$R^2 = 0.65$; CV = 0.30; Predict (20) = 50%; Range = 241 - 10,976 MM [8-3]
where dummy variable $D_1 = 1$ for programs with %Reuse > 39% and 0 otherwise

$$\text{Schedule (Months)} = 5.01 * (\text{MM})^{0.3205} * e^{(-0.5580 * D_1)}$$

$R^2 = 0.63$; CV = 0.31; Predict (20) = 56%; Range = 241 - 10,976 MM [8-4]
where dummy variable $D_1 = 1$ for programs with KSLOC > 30 and 0 otherwise

Non-Traditional Equations: Actual Schedule = f(Estimated ESLOC)

$$\text{Schedule (months)} = 2.16 * (\text{ESLOC})^{0.2270}$$

$R^2 = 0.28$; CV = 0.42; Predict (20) = 44%; Efactor = 0; Range = 20 - 557 KSLOC [8-5]

$$\text{Schedule (Months)} = 1.32 * (\text{ESLOC})^{0.2439} * e^{(0.5389 * D_1)}$$

$R^2 = 0.52$; CV = 0.35; Predict (20) = 50%; Efactor = 0.52; Range = 20 - 557 KSLOC [8-6]
where dummy variable $D_1 = 1$ for programs 100% new and 0 otherwise

Equations [8-3] and [8-4] had the best statistical attributes of the six equations. However, upon a more detailed analysis of equation [8-3]'s residuals, it appears that the regression is biased. It consistently overestimated the very small and very large points, and consistently underestimated those in between. Thus, this equation was eliminated from further consideration. Equation [8-4]'s dummy variable is based on only two data points (i.e., there are only two data points with SLOC less than 30,000), so it was also eliminated from further consideration.

Of the remaining four equations, equation [8-2], the traditional schedule estimating equation with a 100 percent dummy variable had the best statistical attributes. A review of the residuals uncovered no apparent bias. Furthermore, equation [8-2] generates estimates that are comparable to those produced by other traditional schedule estimation models (e.g., COCOMO, REVIC, etc.). Table 8-4 compares the equation [8-2] estimate and several other models' estimates for a 100 man-month effort. The average schedule estimate for the first 11 models in this example is 15 months. The schedule estimates ranged from 10 to 21 months. Barry Boehm's latest COCOMO II schedule equation, which incorporates the latest software methods (unlike the first 11), estimates the schedule for the 100 man-months effort between 14 and 17 months. Equation [8-2] resulted in an estimate of 15 months (assuming the program is not 100 percent new) or 21 months (assuming the program is 100 percent new). These results were comparable to the range of estimates generated by the other estimating models. Additionally, equation [8-2] captured one of the significant drivers (i.e., code condition) highlighted in Section 4 - **Effort Analysis: Significant Drivers**.

| Reference | Equation Schedule | Months ⁵³ |
|--------------------------------|---|----------------------|
| Freburger and Basili, 1979 | $TDEV = 4.38 * (MM)^{0.25}$ | 14 |
| COCOMO: Embedded Mode | $TDEV = 2.5 * (MM)^{0.32}$ | 11 |
| Putnam, 1978: Minimal Schedule | $TDEV = 2.15 * (MM)^{0.333}$ | 10 |
| COCOMO: Semi-detached Mode | $TDEV = 2.5 * (MM)^{0.35}$ | 13 |
| Walston and Felix, 1977 | $TDEV = 2.47 * (MM)^{0.35}$ | 12 |
| Nelson, 1978 | $TDEV = 3.04 * (MM)^{0.36}$ | 16 |
| COCOMO: Organic Mode | $TDEV = 2.5 * (MM)^{0.38}$ | 14 |
| REVIC: Ada Mode | $TDEV = 4.376 * (MM)^{0.32}$ | 19 |
| REVIC: Organic Mode | $TDEV = 3.65 * (MM)^{0.38}$ | 21 |
| REVIC: Semi-detached Mode | $TDEV = 3.8 * (MM)^{0.35}$ | 19 |
| REVIC: Embedded Mode | $TDEV = 4.376 * (MM)^{0.32}$ | 19 |
| | Average = | 15 |
| COCOMO 2.0 | $TDEV = [3.0 * (MM)^{(0.33 + 0.2 * (B - 1.01))}] * (\%Schedule/100)^{54}$ | 14 to 17 |
| Equation [8-2] | $TDEV = 5.12 * (MM)^{0.2266} * e^{(D_1 * 0.3574)}$ | 15 or 21 |

Table 8-4: Comparison of Equation [8-2] to Other Traditional Schedule Estimation Models

8.4 RECOMMENDATIONS

The factors derived from Partition 5 in Table 8-3 and the traditional equation [8-2] have similar statistics, but the traditional equation [8-2] is slightly better. This fact combined with the insensitivity of the factors (i.e., they estimate the same schedule for a program with effort equal to 2559 man-months as they do for one equal to 4000 man-months), leads NCCA to recommend using equation [8-2]:

$$\text{Schedule (Months)} = 5.12 * (MM)^{0.2266} * e^{(0.3574 * D_1)} \quad [8-2]$$

$R^2 = 0.64$; $CV = 0.31$; $\text{Predict (20)} = 44\%$; $\text{Range} = 157 - 10,976 \text{ MM}$
where dummy variable $D_1 = 1$ for programs 100% new and 0 otherwise

⁵³Estimate based on effort estimate of 100 man-months.

⁵⁴Where B = 1.01 or 1.26 based on the COCOMO 2.0 documentation.

Figure 8-2 illustrates the software schedule estimating process.

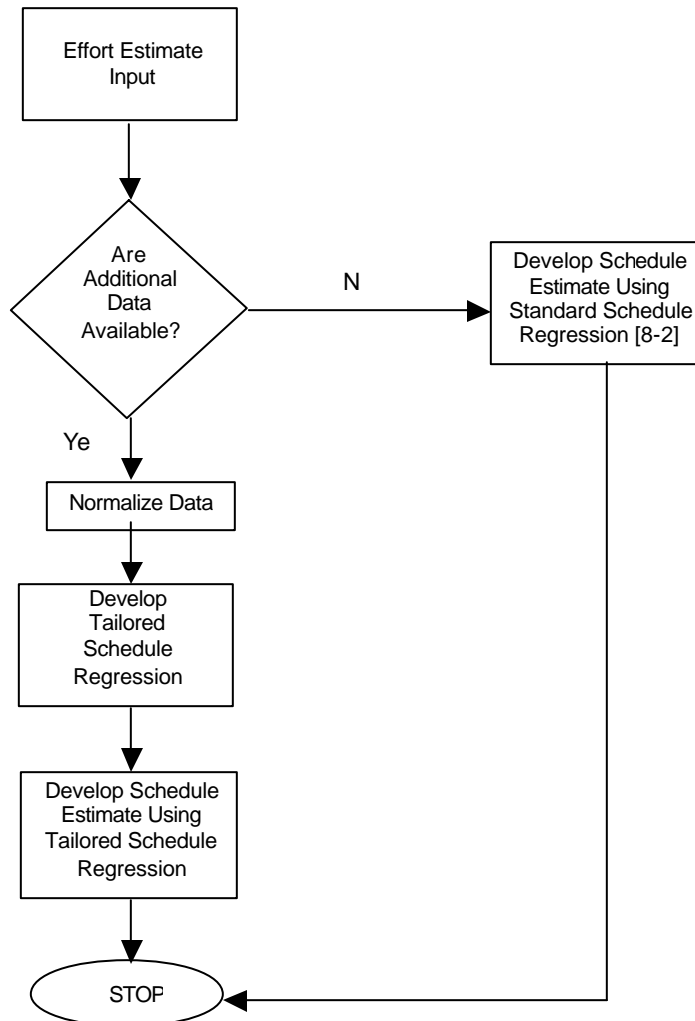


Figure 8-2: Recommended Software Schedule Estimating Process

8.5 CONCLUSIONS

A schedule estimate is an important aspect in developing a software estimate because it impacts both the effort required and the program risk.⁵⁵ In this section, two schedule estimating methodologies (factors and estimating relationships) were developed to estimate schedule. These tools are based on a relatively small database of programs from one data source. Therefore, they may not reflect an industry average and should be used with caution. Although these factors and equations were developed from a small database, the statistics associated with the recommended tool are satisfactory and the underlying database represents a wide range of program sizes. Because software schedules are often influenced by external factors such as budget cuts or requirements creep, they will remain difficult to estimate. However,

⁵⁵Refer to the NCCA Issue Paper "The Impacts of Schedule Compression/Slippage/Stretch-Out on the Software Development Schedule" for a discussion of this topic.

NCCA has identified several parameters which impact schedule, and which should be considered when developing a software development schedule estimate. As with the effort analysis, NCCA's recommendation is that this methodology be used only when contractor-specific data, appropriate for developing tailored equations, is not available.

8.6 FUTURE EFFORTS

A future goal is to collect additional data to improve schedule regressions. The NCCA Normalized Schedule Database would be improved if the following data were collected:

- 1) Elapsed time between effort phases.
- 2) Dates for software-related reviews and phases.
- 3) MIS program schedule data.

LABOR RATE ANALYSIS

9.1 INTRODUCTION

This section of the handbook delineates how NCCA collected, analyzed, and normalized Cost Performance Report (CPR) and Contractor Cost Data Report (CCDR) data for use in developing an effort-to-cost/price estimating methodology and tool. The methodology used to develop the effort-to-cost estimating tool is described in the following subsections:

- NCCA Labor Rate Databases
- Methodology and Results
- Recommendations
- Conclusions
- Additional Considerations
- Future Efforts

9.2 NCCA LABOR RATE DATABASES

NCCA also created a separate labor rate database to support the development of software specific man-year rate analyses. The detailed methodology will be discussed below.

9.2.1 GROUND RULES AND ASSUMPTIONS

Provided below are the ground rules and assumptions utilized to develop the NCCA Labor Rate Databases:

- 1) NCCA retained only those programs (or data points) that were at least 90 percent complete. It was assumed that a program, which has expended 90 percent of its estimate at completion (as portrayed in the CCDR forms) is at least 90 percent complete.
- 2) The man-hours of effort reported for the EMD Phase covered the software development life cycle, defined as SDR through FQT (See Figure 3-3).
- 3) For CPR data points that reported effort in man-months, NCCA assumed a 152-hours per man-month rate for the conversion from man-months to hours.
- 4) Since DoD-STD-2167A went into effect in 1985, NCCA made the assumption that contracts awarded prior to 1985 were pre-DoD-STD-2167A and contracts awarded in 1985 and after were post-DoD-STD-2167A. There is the possibility, though, that a contractor received a waiver to utilize a previous standard after 1985.
- 5) The software costs expended were assumed to be then-year dollars based on the year of the contract. For example, if program X's contract number was N00019-88-C-0001, then


Section 9 – Labor Rate Analysis

program X's software cost was assumed to be in then-year 1988 dollars. All labor hours were normalized to thousands of hours and all costs were normalized to constant FY97\$K, using NCCA's RDT&EN (Purchases) weighted indices, dated March 1996.

9.2.2 DATA SOURCES

NCCA collected software cost data from CCDRs and CPRs, which provided actual cost in addition to other data, for the NCCA Raw Labor Rate Database. Figure 9-1 is an example of a CCDR 1921 form, which reports cost-to-date, as well as, cost-at-completion estimates. Figure 9-2 is an example of a CCDR 1921-1 form, which reports man-hours by WBS element to date and at completion by functional category. NCCA used the **shaded** elements from Figures 9-1 and 9-2 for the labor rate analysis. The most essential elements were the estimated software cost at completion, located on both Figures 9-1 and 9-2, and the total software labor hours estimated at completion found in Figure 9-2. To cross-check, NCCA ensured the 1921 software at completion cost was equal to the 1921-1 software at completion cost.

A CPR reports cost by WBS element. Format 1 of a CPR (Figure 9-3) reports the top-level cost for the program by WBS element. Format 4 of a CPR (Figure 9-4) reports man-months expended to date and at completion. The **shaded** elements in Figures 9-3 and 9-4 are inputs into the database. The most essential elements are the software's latest revised estimate (LRE) at completion in Figure 9-3 and the estimate at completion man-months for software in Figure 9-4.

| CONTRACT COST DATA REPORT (CCDR) (DOLLARS IN THOUSANDS) | | 1. PROGRAM/CONTRACT NO.: EX A/C/N00019-89-C-6083 | | 2. (X) CONTRACT () RFP () PROGRAM EST | | 3. (X) RDT&E () PROD 4. MULTIPLE CONTRACT (X) YES () NO | | 5. REPORT AS OF 30 JUNE 1989 6. FY FUNDED: 1990 | |
|---|-------------------------|---|---------------------------|---|---|---|---------------------------------|---|------------|
| 7. CONTRACT TYPE: CPIF | | 8. CONTRACT PRICE 208,847 | | 9. CONTRACT CEILING 210,000 | | 10. (X) PRIME () SUB XYZ, INC. XYZ, VIRGINIA | | 11. NAME OF CUSTOMER (SUBCNTR USE ONLY) | |
| CONTRACT LINE ITEM A | REPORTING ELEMENTS B | ELEMENT CODE C | TO DATE COSTS INCURRED | | | UNITS G | AT COMPLETION COSTS INCURRED | | |
| | | | | | | | | | |
| | | | NON- RECURRING D | RECURRING E | TOTAL F | | NON- RECURRING H | RECURRING I | TOTAL J |
| 100 | H/W | 2 | 94000 | 0 | 94000 | | 101000 | 0 | 101000 |
| 200 | S/W | 2 | 80000 | 0 | 80000 | | 85000 | 0 | 85000 |
| 300 | LS | 2 | 930 | 0 | 930 | | 1200 | 0 | 1200 |
| | MANUFACTURING COST | | 174930 | 0 | 174930 | | 187200 | 0 | 187200 |
| | G&A | | 17493 | 0 | 17493 | | 18720 | 0 | 18720 |
| | COM | | 2099 | 0 | 2099 | | 2246 | 0 | 2246 |
| | FEE | | 19242 | 0 | 19242 | | 20592 | 0 | 20592 |
| | TOTAL PRICE | | 213764 | 0 | 213764 | | 228758 | 0 | 228758 |
| REMARKS | | | | | | | | | |
| NAME OF PERSON TO BE CONTACTED JOE SMITH, MANAGER | | | | | SIGNATURE  | | | DATE 3 JUL 89 | |

DD FORM 1921

GO7011-02

Figure 9-1: Contractor Cost Data Report, 1921

Section 9 – Labor Rate Analysis

| FUNCTIONAL COST-HOUR REPORT | | 1. PROGRAM: EX A/C | | 2. REPORT AS OF 30 JUNE 1989 | | | |
|---|--|--|-----------------|---|----------|-----------|------------|
| 3. DOLLARS IN THOUSANDS | | 4. HOURS IN THOUSANDS | | 5. (X) CONTRACT () PROGRAM EST () RFP | | | |
| 6. () NON-RECURRING () RECURRING (X) TOTAL | | | | 7. (X) RDT&E () PROD () OTHER | | | |
| 8. MULTIPLE YEAR CONTRACT (X) YES () NO | | 10. (X) PRIME () SUBCONTRACTOR XYZ, INC. XYZ, VIRGINIA | | 11. NAME OF CUSTOMER (SUBCNTR USE ONLY) | | | |
| 9. FY FUNDED 1990 | | | | | | | |
| 12. REPORTING ELEMENT(S) 200 SOFTWARE | | | | | | | |
| FUNCTIONAL CATEGORIES | ADJUST- MENTS TO PREVIOUS REPORTS | CONTRACTOR | | SUBCONTRACT OR OUT- SIDE PROD AND SERV | | TOTAL | |
| | | TO DATE | AT COMPL | TO DATE | AT COMPL | TO DATE | AT COMPL |
| | A | B | C | D | E | F | G |
| ENGINEERING | | | | | | | |
| 1. DIRECT LABOR HOURS | | 100 | 5000 | | | 4600 | 5000 |
| 2. DIRECT LABOR DOLLARS | \$ | \$ 500 | \$ 20,000 | \$ | \$ | \$ 19,250 | \$ 20,000 |
| 3. OVERHEAD | \$ | \$ 1400 | \$ 45,000 | \$ | \$ | \$ 44,500 | \$ 45,000 |
| 4. MATERIAL | \$ | \$ 100 | \$ 5,000 | \$ | \$ | \$ 4,750 | \$ 5,000 |
| 5. OTHER DIRECT CHARGES (Specify) | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 6. TOTAL ENGINEERING DOLLARS | \$ | \$ 2,000 | \$ 70,000 | \$ | \$ | \$ 68,500 | \$ 70,000 |
| TOOLING | | | | | | | |
| 7. DIRECT LABOR HOURS | | 1699 | 2000 | | | 1699 | 2000 |
| 8. DIRECT LABOR DOLLARS | \$ | \$ 3,940 | \$ 4,000 | \$ | \$ | \$ 3,940 | \$ 4,000 |
| 9. OVERHEAD | \$ | \$ 5,550 | \$ 6,000 | \$ | \$ | \$ 5,550 | \$ 6,000 |
| 10. MATERIAL | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 11. OTHER DIRECT CHARGES (Specify) | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 12. TOTAL TOOLING DOLLARS | \$ | \$ 9,490 | \$ 10,000 | \$ | \$ | \$ 9,490 | \$ 10,000 |
| QUALITY CONTROL | | | | | | | |
| 13. DIRECT LABOR HOURS | | 250 | 364 | | | 250 | 364 |
| 14. DIRECT LABOR DOLLARS | \$ | \$ 300 | \$ 500 | \$ | \$ | \$ 300 | \$ 500 |
| 15. OVERHEAD | \$ | \$ 600 | \$ 1,000 | \$ | \$ | \$ 600 | \$ 1,000 |
| 16. OTHER DIRECT CHARGES (Specify) | \$ | \$ 25 | \$ 0 | \$ | \$ | \$ 25 | \$ 0 |
| 17. TOTAL QUALITY CNTRL DOLLARS | \$ | \$ 925 | \$ 1,500 | \$ | \$ | \$ 925 | \$ 1,500 |
| MANUFACTURING | | | | | | | |
| 18. DIRECT LABOR HOURS | | 650 | 1000 | | | 650 | 1000 |
| 19. DIRECT LABOR DOLLARS | \$ | \$ 600 | \$ 1,000 | \$ | \$ | \$ 600 | \$ 1,000 |
| 20. OVERHEAD | \$ | \$ 1,000 | \$ 2,500 | \$ | \$ | \$ 1,000 | \$ 2,500 |
| 21. MATRL AND PURCHASED PARTS | \$ | \$ 485 | \$ 0 | \$ | \$ | \$ 485 | \$ 0 |
| 22. OTHER DIRECT CHARGES (Specify) | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 23. TOTAL MANUFACTURING DOLLARS | \$ | \$ 2,085 | \$ 3,500 | \$ | \$ | \$ 2,085 | \$ 3,500 |
| 24. PURCHASED EQUIPMENT | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 25. MATERIAL OVERHEAD | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 26. Other Costs Not Shown Elsewhere | \$ | \$ 0 | \$ 0 | \$ | \$ | \$ 0 | \$ 0 |
| 27. TOTAL COST LESS G&A | \$ | \$ 81,000 | \$ 85,000 | \$ | \$ | \$ 81,000 | \$ 85,000 |
| 28. G&A | \$ | \$ 8,100 | \$ 8,500 | \$ | \$ | \$ 8,100 | \$ 8,500 |
| 29. TOTAL COST PLUS G&A | \$ | \$ 89,100 | \$ 93,500 | \$ | \$ | \$ 89,100 | \$ 93,500 |
| 30. FEE OR PROFIT/COM | \$ | \$ 9,882 | \$ 10,370 | \$ | \$ | \$ 9,882 | \$ 10,370 |
| 31. TOTAL OF LINES 29 AND 30 | \$ | \$ 98,982 | \$ 103,870 | \$ | \$ | \$ 98,982 | \$ 103,870 |
| DIRECT LABOR MAN-HOURS INCURRED THIS REPORT PERIOD | | | | | | | |
| | ENGINEERING | TOOLING | QUALITY CONTROL | MANUFACTURING | | | |
| | A | B | C | D | | | |
| 1. TOTAL BEG OF PERIOD | | | | | | | |
| 2. | | | | | | | |
| 3. | | | | | | | |
| 4. | | | | | | | |
| 5. | | | | | | | |
| 6 TOTAL END OF PERIOD | | | | | | | |

DD FORM 1921-1

GO7011-02

Figure 9-2: Functional Cost-Hour Report, 1921-1

Section 9 – Labor Rate Analysis

| Cost Performance Report - Work Breakdown Structure | | | | | | | | | | | | | | |
|--|---------------|-------------------------|---------------------|----------------------|---------------|-----------------------------|---------------------|--------------------------|--------|---------|-------------------------------|----------|--------|----------|
| Contractor: XYZ, INC. | | Contract Type/No.: CPIF | | Program Name: EX A/C | | Report Period: 30 JUNE 1989 | | Signature, Title & Date: | | | FORM APPROVED BY: | | | |
| Location: XYZ, VA | | | | | | | | | | | OMB NUMBER 0000000000000000 | | | |
| RDT&E (X) PROD () | | N00018-88-C-6083 | | | | | | | | | | | | |
| Quantity | Negot. Cost: | Est Cost Auth | TGT PRFT/ FEE PCT | TGT Price: | Est Price: | Share Ratio: | Contract Ceiling: | Est Ceiling: | | | | | | |
| | 209,110 | 0 | 11% | 0 | 0 | 0%/100% | 210,000 | 0 | | | | | | |
| WBS EX/AC | CURRENT | | | | | CUMULATIVE TO DATE | | | | | AT COMPLETION | | | |
| | BUDGETED COST | | ACTUAL COST WORK | VARIANCE | BUDGETED COST | | ACTUAL COST WORK | VARIANCE | | BUDGET | LATEST REVISED ESTIMATE | VARIANCE | | |
| | WORK | WORK | | | WORK | WORK | | PERF | PERF | | | | | |
| | SCHED | PERF | PERF | SCHED | COST | SCHED | PERF | PERF | SCHED | | | | COST | |
| WBS | | | | | | | | | | | | | | |
| 100 H/W | | 2502 | 1349 | 3172 | -1153 | -1823 | 100000 | 96000 | 94000 | -4000 | 2000 | 96500 | 101000 | 4500 |
| 200 S/W | | 3000 | 2000 | 3500 | -1000 | -1500 | 82000 | 81000 | 80000 | -1000 | 1000 | 75500 | 85000 | 9500 |
| 300 ILS | | 200 | 100 | 250 | -100 | -150 | 1000 | 950 | 930 | -50 | 20 | 975 | 1200 | 225 |
| FACTORY COST | | 5702 | 3449 | 6922 | -2253 | -3473 | 183000 | 177950 | 174930 | -5050 | 3020 | 172975 | 187200 | 14225 |
| COST OF MONEY | | 68 | 41 | 83 | -27.036 | -42 | 2196 | 2135 | 2099 | -60.6 | 36.24 | 2076 | 2246 | 170.7 |
| GEN AND ADMIN | | 570 | 345 | 692 | -225.3 | -347 | 18300 | 17795 | 17493 | -505 | 302 | 17298 | 18720 | 1422.5 |
| UNDISTRIBUTED BUDGET | | | | | | | | | | | | | | |
| SUBTOTAL | | 6341 | 3835 | 7697 | -2505 | -3862 | 203496 | 197880 | 194522 | -5616 | 3358 | 192348 | 208166 | 15818 |
| PROFIT | | 627 | 379 | 761 | -248 | -382 | 20130 | 19575 | 19242 | -556 | 332 | 19027 | 20592 | 1565 |
| UNASSIGNED FUNDS | | | | | | | | | | | | | | |
| TOTAL | | 6968 | 4215 | 8459 | -2753.166 | -4244 | 223626 | 217455 | 213764 | -6171.1 | 3690.4 | 211375 | 228758 | 17382.95 |
| | | | | | | | | | | | | | | |
| (DOLLARS IN THOUSANDS) | | | | | | | | | | | | | | |
| FORMAT 1 | | | | | | | | | | | | | | |

Figure 9-3: Cost Performance Report, Format 1

| COST PERFORMANCE REPORT - MANPOWER LOADING | | | | | | | | | | | | | |
|---|-----------------------------|--|-----------------------------|----------------------|-----|-----|-----|-----------------------------|---------------------------|---|---------|-------|------------------|
| CONTRACTOR: XYZ, INC | | CONTRACT TYPE/NUMBER: CPIF | | PROGRAM NAME: EX A/C | | | | REPORT PERIOD: 30 JUNE 1989 | | FORM APPROVED BY: OMB NUMBER 0000000000000000 | | | |
| LOCATION: XYZ, VA | | | | | | | | | | | | | |
| RDT&E (X) PROD () | | N00018-88-C-6083 | | | | | | | | | | | |
| ORGANIZATIONAL OR FUNCTIONAL CATEGORY | ACTUAL CURRENT PERIOD | ACTUAL END OF CURRENT PERIOD (CUM) | FORECAST (NON-CUMULATIVE) | | | | | | | | | | |
| | | | SIX MONTH FORECAST BY MONTH | | | | | | (Enter Specified Periods) | | | | AT COMPLETION |
| | | | (ENTER NAMES OF MONTH) | | | | | | | | | | |
| | | | MAR | APR | MAY | JUN | JUL | AUG | EOY87 | CY88 | 1989-90 | | |
| 100 H/W | 31 | 3475 | 62 | 23 | 17 | 11 | 6 | 11 | 51 | 28 | 0 | 3684 | |
| 200 S/W | 241 | 7199 | 211 | 156 | 148 | 162 | 98 | 82 | 179 | 121 | 8 | 8364 | |
| 300 ILS | 36 | 719 | 20 | 15 | 12 | 12 | 12 | 11 | 59 | 160 | 48 | 1068 | |
| TOTAL DIRECT | 308 | 11393 | 293 | 194 | 177 | 185 | 116 | 104 | 289 | 309 | 56 | 13116 | |
| ALL FIGURES IN WHOLE NUMBERS | | | | | | | | | | | | | |
| FORMAT 4 | | | | | | | | | | | | | |

Figure 9-4: Cost Performance Report Manpower Loading Report, Format 4

9.2.3 RAW LABOR RATE DATABASE

NCCA collected cost and manning data for 34 software efforts. The software cost data was in then-year dollars, including G&A, and the manning data is in man-hours or man-months. Table 9-1 lists all of the data points collected for the software labor rate analysis. These data points met the initial criterion for database inclusion, which means they reported software cost. After scrutiny of the data points, some programs, as shown in the shaded areas of Table 9-1, were excluded from the database for one of the following reasons:

- 1) Man-hours were not reported.
- 2) Software development effort did not include the requirements phase.
- 3) Software development effort experienced major problems, (i.e., flight test failures).
- 4) Programs were less than 90 percent complete.

| | Program | Labor Hours | Status of Completion |
|----|-----------------|---------------------------------|-------------------------------|
| 1 | A-6E Upgrade | not provided | |
| 2 | AAAM | not provided | |
| 3 | AAAM | Provided | |
| 4 | AAAM | Provided | |
| 5 | AAAM (D&V) | not provided | |
| 6 | ALFS | provided | Less than 90 percent complete |
| 7 | AMRAAM AIM-120A | not provided | |
| 8 | AMRAAM SYS | provided | |
| 9 | AN/ALR-67 | provided | Less than 90 percent complete |
| 10 | AN/ALR-77 | not provided | |
| 11 | APG-71 F-14D | provided | |
| 12 | APG-73 RADAR | provided | |
| 13 | AQM-127A | provided, s/w failure | |
| 14 | ASPJ | provided, effort < requirements | |
| 15 | ASPJ | provided, effort < requirements | |
| 16 | BSY-1 | provided | Less than 90 percent complete |
| 17 | BSY-1 | provided | |
| 18 | BSY-1 | provided | |
| 19 | BSY-2 | provided | |
| 20 | CASS | provided | |
| 21 | CEC | Provided | |
| 22 | E-2C GRP2, PT2 | not provided | |
| 23 | ES-3A PROTOTYPE | Provided | |
| 24 | F-14A & F-14D | Provided | |
| 25 | F-18 FSD | not provided | |
| 26 | F/A-18 OTPS | Provided | Less than 90 percent complete |
| 27 | HARM-CLCP | Provided | |
| 28 | JSOW | Provided | |
| 29 | MHIP | Provided | |
| 30 | P-3UPD IV | Provided | Less than 90 percent complete |
| 31 | S-3A | not provided | |
| 32 | S-3B | not provided | |
| 33 | SMIP LOW | not provided | |
| 34 | SPAR/AIM/RIM-7P | not provided | |

Table 9-1: NCCA Raw Software Labor Rate Database

9.2.4 NCCA NORMALIZED LABOR RATE DATABASE

The final sanitized,⁵⁶ normalized software NCCA Labor Rate Database has 15 programs with 10 data fields as shown in Table 9-2. The man-hours expended to develop the software range from 2K to 793K labor hours and the software costs through G&A range from \$317K to \$95M in constant FY97\$. The NCCA Normalized Labor Rate Database's population consists of aircraft, ships, missiles and electronics programs, representing cost-plus and fixed-price contracts. The East Coast contractors are located from Rhode Island to Florida. The West Coast contractors are located from California to Texas. The first year of development ranged from 1982 to 1992. NCCA did not collect MIS program data and all programs except one were embedded. Each data field, shown in Table 9-2, is defined below.

| Program | Contractor | Contractor Location | Contract Type | DoD-Std 2167A | Platform | Labor KHrs | Total Cost (\$K) | \$/Hr | %Expd/ Compl |
|---------|------------|---------------------|---------------|---------------|----------|------------|------------------|----------|--------------------|
| NCCA 1 | 1 | West | CPIF | Post | Missile | 49.90 | \$4,709.62 | \$94.38 | 88.3 ⁵⁷ |
| NCCA 2 | 2 | East | CPIF | Post | Missile | 63.88 | \$7,170.56 | \$112.25 | 95.2 |
| NCCA 3 | 3 | West | FPI | Pre | Missile | 560.09 | \$45,164.53 | \$80.64 | 99.8 |
| NCCA 4 | 3 | West | FFP | Pre | Aircraft | 792.80 | \$86,637.84 | \$109.28 | 97.3 |
| NCCA 5 | 3 | West | FFP | Post | Aircraft | 314.22 | \$41,037.60 | \$130.60 | 100.0 |
| NCCA 6 | 4 | East | CPIF | Pre | Ship | 345.95 | \$25,235.64 | \$72.95 | 92.1 |
| NCCA 7 | 5 | West | CPIF | Pre | Ship | 207.75 | \$16,490.62 | \$79.38 | 98.5 |
| NCCA 8 | 6 | West | FPI | Post | Ship | 652.10 | \$57,282.64 | \$87.84 | 100.0 |
| NCCA 9 | 7 | East | FFP | Post | Elx | 138.00 | \$9,571.33 | \$69.36 | 99.7 |
| NCCA 10 | 8 | East | CPAF/FF | Post | Ship | 326.69 | \$26,067.94 | \$79.79 | 99.3 |
| NCCA 11 | 9 | East | FPI | Post | Aircraft | 2.41 | \$316.59 | \$131.64 | 100.0 |
| NCCA 12 | 10 | East | FFP | Pre | Aircraft | 723.99 | \$94,602.96 | \$130.67 | 91.9 |
| NCCA 13 | 11 | West | FFP | Post | Missile | 11.74 | \$911.64 | \$77.69 | 98.7 |
| NCCA 14 | 11 | West | CPIF | Post | Missile | 202.26 | \$17,902.27 | \$88.51 | 90.1 |
| NCCA 15 | 3 | West | CPFF | Post | Missile | 115.00 | \$10,372.80 | \$90.20 | 96.6 |

Table 9-2: NCCA Normalized Software Labor Rate Database FY97\$K (Cost through G&A)

1) **Program:** Records the program name for each software development. **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Labor Rate Database.** See Appendix F for this information.

2) **Contractor:** Records the name of the contractor for identification purposes and for development of contractor specific labor rate databases. **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Labor Rate Database.** See Appendix F for this information.

3) **Contractor Location:** Records the city and state of the facility for each program. After all the locations were collected, NCCA categorized the states into two regions (east or west). **This information is business sensitive and is withheld from the sanitized version of the NCCA Raw Labor Rate Database.** See Appendix F for this information.

4) **Contract Type:** Records the contract type for each software development effort. The database includes cost-plus and fixed-price contract types.

⁵⁶Due to the proprietary nature of the data, program name, developing contractor name and labor rate will not be published in tandem. The key code to the data points is provided in Appendix F.

⁵⁷NCCA made an exception to the 90 percent or more rule for this program because it had two prime contractors. One contractor was slightly less than 90 percent expended, but the combined percent expended for both contractors was 92 percent.

5) **DoD-STD-2167A:** Records the development standard. NCCA used the contract number to determine the date of contract award. Based on this year, NCCA determined if the development effort was prior to or after DoD-STD-2167A was implemented. Prior to 1985, other standards were used to develop software.

6) **Platform:** Records the platform type for each program (aircraft, missiles, ships or electronics).

7) **Labor KHours:** Records the total hours expended to develop software for the program. The CCDR reports effort in hours and the CPR reports effort in man-months.

8) **Total Cost (\$K):** Records the cost at the total software level in thousands. The total software cost level is the total amount of money expended for developing the program software. The total software cost includes both direct and indirect costs. It does not include COM and Fee/Profit; these burden rates are included in the price level analysis.

9) **\$/Hr:** Total dollars expended divided by total labor hours for the program's software effort.

10) **%Expd/Compl:** Calculates the percent complete. This equates to the actual software cost spent to date on a CCDR divided by the total software cost at completion. For CPRs, NCCA calculated the percent complete by dividing the software actual cost of work performed (ACWP) by the software LRE.

9.3 METHODOLOGY AND RESULTS

There are several approaches that can be used to convert software effort to software cost. The preferred approach, if the contractor is known, is to utilize the Forward Pricing Rate Agreements (FPRAs). These rates are the actual Defense Contracting Audit Agency (DCAA) approved or recommended rates for the contractor; they best represent current and future business base conditions. However, to apply these rates correctly, not only does the analyst require a proposed or representative historical software team composition, but the analyst also should be aware of how the historical team composition compares with the proposed team composition for the program being estimated. Appendix F provides a detailed example and further direction on the correct application of FPRAs. If the FPRAs are not available, NCCA recommends the analyst try to develop a contractor-specific labor rate database based on the most analogous data available. Appendix F provides detailed procedures for the application of a contractor-specific labor rate. Both approaches focus on retrieving the most analogous historical data available to estimate the software cost. However, if the analyst has been unsuccessful in performing either approach, NCCA recommends the analyst utilize the software team's effort-to-cost conversion tool discussed in this section.

In the process of developing an effort-to-cost conversion tool, NCCA computed average labor rates, performed nonparametric analyses using the Wilcoxon Two-Sample test and ran both linear and power regressions. Technical information such as sizing, code condition and language was not provided; therefore, the productivity drivers identified in Section 4 - **Effort Analysis: Significant Drivers** could not be used to partition the database. The other characteristics (contract type, platform type, contractor location, number of labor hours and

software development standard) were provided for each program and were investigated as database partitions using the non-parametric Wilcoxon Two-Sample test. These characteristics were examined as database partitions for the following reasons:

Contract Type

NCCA attempted to capture the different rate structures due to the inherent risks of different contract types. A fixed price contract places full assumption of risk, cost, and profit or loss on the contractor. A cost plus contract is a total cost reimbursement contract where the government is required to reimburse the contractor for all reasonable and allocable costs incurred during contract performance. There were seven cost plus data points and eight fixed price data points.

Platform Type

NCCA attempted to capture the difference between platforms; air systems have greater physical constraints than non-air systems. The database consisted of software development efforts for aircraft avionics, missiles, shipboard electronics and ground electronics programs. There were four aircraft related data points and eleven non-aircraft (missiles, ships, electronic) data points.

Contractor Location

NCCA divided the database to investigate possible geographical (i.e., east versus west) differences in rate structures due to varying cost of living levels. The database included nine contractors in the west (California, Texas, Arizona) and six contractors on the East Coast (Rhode Island, Maryland, Virginia, Florida, New Jersey).

Number of Labor Hours

Graphically, it appeared that there were two separate data sets and the database breakpoint was at a size of 200K. Six data points had less than 200K labor hours and nine data points had more than 200K labor hours.

Software Development Regulation

In the database, the years of software development range from 1982 through 1992. During this ten year span, there were different DoD standards used for different programs. After 1985, DoD-STD-2167A was the standard for developing software. Prior to 1985, there were several DoD standards utilized for software development. Each standard had different requirements for documenting the process, controlling the process, presenting the process, and conducting reviews of the process. Five programs in the database were developed prior to DoD-STD-2167A and ten programs were developed after DoD-STD-2167A became effective.

9.3.1 AVERAGE LABOR RATE ANALYSIS

Average labor rates were developed for the database partitions described above. NCCA calculated the labor rates by dividing software dollars by software labor hours.

Section 9 – Labor Rate Analysis

Table 9-3 summarizes the results and statistics from the average rate analysis. The back-up spreadsheets for the average rate analysis are provided in Appendix F.

| | Population | Avg Rate/Hr (FY97\$) | SEE ⁵⁸ | CV _{est} ⁵⁹ |
|----------------------------------|--------------------|----------------------|-------------------|---------------------------------|
| Average Rate One | Total (Top-Level) | \$95.68 | 9167.90 | 31% |
| Average Rate Two (Contract Type) | Cost Plus | \$88.21 | 7995.93 | 27% |
| | Fixed Price | \$102.21 | | |
| Average Rate Three (Platform) | Aircraft | \$125.55 | 4128.69 | 14% |
| | Non-Aircraft | \$84.82 | | |
| Average Rate Four (Location) | East | \$99.44 | 8603.82 | 29% |
| | West | \$93.17 | | |
| Average Rate Five (Size) | <200K Hours | \$95.92 | 9191.90 | 31% |
| | >200K Hours | \$95.52 | | |
| Average Rate Six (Standard) | Pre DoD-STD-2167A | \$94.58 | 9358.48 | 32% |
| | Post DoD-STD-2167A | \$96.23 | | |

Table 9-3: Average Labor Rate Analysis (Cost through G&A)

To complete the average labor rate analysis, the Wilcoxon Two-Sample test was performed for each population to determine if the population means were statistically different, where H_0 , the null hypotheses, assumes that the population means are equal. Table 9-4 presents the final results from each test (see Appendix F). Based on the results from the Wilcoxon Two-Sample test (see Appendix C), NCCA concluded that platform type was the only significant driver among the five variables examined. Since the means were proven to be statistically different, an aircraft software development estimate should use an aircraft specific average labor rate, while ship, missile, and electronic software development estimates should use the non-aircraft average labor rate. Since platform type was the only significant driver, it was the only dummy variable used in the follow-on regression analyses.

| | Population | Reject H_0 |
|------------|---|--------------|
| Test One | Cost Plus vs. Fixed Price | No |
| Test Two | Aircraft vs. Non-Aircraft | Yes |
| Test Three | East Coast vs. West Coast | No |
| Test Four | Labor Hours < 200K vs. Labor Hours > 200K | No |
| Test Five | Pre-DoD-STD-2167A vs. Post-DoD-STD-2167A | No |

Table 9-4: Nonparametric Analysis

9.3.2 REGRESSION ANALYSIS

In an attempt to improve the statistics of the average labor rates shown above, NCCA performed linear and power regression analyses. Software cost through G&A (FY97\$K) was the dependent variable. Thousands of labor hours and the platform dummy variable, or dummy slope, were the independent variables. For both sets of analyses (linear and power), the methodology for regression analyses sets one ($\$ = f(\text{labor hours})$) and two ($\$ = f(\text{labor hours and platform type})$) was the same. The final spreadsheets for each analysis are provided in Appendix F and the analyses and results are detailed below.

⁵⁸ SEE is the standard error of the estimate is a measure of the deviation of the sample data points from the regression line.

⁵⁹ $CV_{est} = \frac{SEE_{dataset}}{\bar{Y}}$, where $\bar{Y} = \$29,564.97$.

9.3.2.1 REGRESSION ANALYSIS SET ONE

NCCA regressed software dollars as a function of labor hours. This analysis provided both linear and power regressions, equations [9-1] and [9-2], for estimating the development cost of software. The Labor Khrs variable was significant at the 95 percent confidence level.

$$\text{FY97\$K} = -3,045.36 + 108.54 * (\text{Labor Khrs})$$

$R^2 = 0.93$; $CV = 0.28$; Predict (20) = 33%; Range 2.41 - 792.80 Labor Khours⁶⁰ [9-1]

$$\text{FY97\$K} = 103.26 * (\text{Labor Khrs})^{0.98}$$

$R^2 = 0.98$; $CV = 0.22$; Predict (20) = 60%; Range 2.41 - 792.80 Labor Khours [9-2]

9.3.2.2 REGRESSION ANALYSIS SET TWO

NCCA regressed software dollars as a function of labor hours and the dummy intercept variable, platform type (equations [9-3] - linear and [9-5] - power). Platform type was also used as a dummy slope variable (equations [9-4] - linear and [9-6] - power). The dummy slope variable assumes that the slopes of the two regression lines are different, but that the intercept terms are identical. Platform type was significant at the 95 percent confidence level.

Equation [9-3] is the linear equation, using the dummy intercept.

$$\text{FY97\$K} = 10,055.89 + 99.47 * (\text{Labor Khrs}) - 14,150.64 * D_1$$

$R^2 = 0.97$; $CV = 0.20$; Predict (20) = 60%; Range 2.41 - 792.80 Labor Khours⁶¹ [9-3]
where $D_1 = 0$ for aircraft and 1 for non-aircraft

Equation [9-4] is the linear equation, using the dummy slope.

$$\text{FY97\$K} = 374.32 + 119.37 * (\text{Labor Khrs}) - 37.45 * [D_1 * \text{Labor Khrs}]$$

$R^2 = 0.99$; $CV = 0.13$; Predict (20) = 73%; Range 2.41 - 792.80 Labor Khours [9-4]
where $D_1 = 0$ for aircraft and 1 for non-aircraft

Equation [9-5] is the power equation, using the dummy intercept.

$$\text{FY97\$K} = 136.98 * (\text{Labor Khrs})^{0.98} * e^{(D_1 * -0.40)}$$

$R^2 = 0.99$; $CV = 0.13$; Predict (20) = 87%; Range 2.41 - 792.80 Labor Khours [9-5]
where $D_1 = 0$ for aircraft and 1 for non-aircraft

Equation [9-6] is the power equation, using the dummy slope.

$$\text{FY97\$K} = 95.91 * (\text{Labor Khrs})^{1.02 - (D_1 * -.0006)}$$

$R^2 = 0.99$; $CV = 0.20$; Predict (20) = 47%; Range 2.41 - 792.80 Labor Khours [9-6]
where $D_1 = 0$ for aircraft and 1 for non-aircraft

⁶⁰ Labor hours less than 28K will result in a negative cost.

⁶¹ Non-aircraft programs with labor hours less than 41K will result in a negative cost.

9.4 RECOMMENDATIONS

Although all six equations developed in Section 9.3.2 were significant, only equations [9-3] through [9-6] capture the impact of the significant driver: platform type. Of these, equations [9-4] and [9-5] had the lowest CVs (13 percent). NCCA then compared equation [9-5] (power form of the equation) to equation [9-4] (the linear form of the equation) to determine how well the power equation performed. Equation [9-5] resulted in smaller residuals than equation [9-4] on small programs; but a majority of the time, the estimates for both equations were within two percent of one another. Equation [9-5] had a Predict (20) of 87 percent in comparison to equation [9-4]'s Predict (20) of 73 percent. A detailed review of the resulting residuals indicated no other trends or biases in the data (i.e., no overestimating or underestimating of large vice small programs, or aircraft vice non-aircraft). For these reasons, equation [9-5] is the recommended regression.

Equation [9-5] was then compared to the average labor rates. For purposes of comparison, the lower level average labor rate (set of factors developed in section 9.3.1) which captures the platform impact is shown below:

$$\text{Aircraft} = \$125.55/\text{Hour}; \text{Non-Aircraft} = \$84.82/\text{Hour} \\ \text{CV}_{\text{est}} = 0.14; \text{Predict (20)} = 87.00\%$$

These two methodologies are essentially identical, however, the CV_{est} of the average labor rate was 14 percent, as shown above. Equation [9-5] has a lower CV (13 percent). However, it should be noted that equation [9-5] has an exponent of 0.98 which indicates a small economy of scale (i.e., it is more cost effective to develop large programs than small programs). This implies that there is a fixed level of cost (possibly captured in overhead) associated with all programs and that, as the hours increased, this cost is allocated across a larger base; hence cost per hour is lower for a larger program. However, because an exponent close to one implies that the relationship is almost linear, the economies of scale are actually quite minute. Therefore, based on the lower CV value, NCCA recommends utilizing the platform specific (aircraft or non-aircraft) equation provided below when analogous or contractor specific data is not available:

$$\text{FY97\$K} = 136.98 * (\text{Labor Khrs})^{0.98} * e^{(D_1 * -0.40)} \quad [9-5] \\ R^2 = 0.99; \text{CV} = 0.13; \text{Predict (20)} = 87\%; \text{Range 2.41 - 792.80 Labor Khours} \\ \text{where } D_1 = 0 \text{ for aircraft and 1 for non-aircraft}$$

Cost of Money (COM) and fee were not available for all data points in the NCCA Normalized Labor Rate Database; therefore, NCCA used what was available to develop average burden rates. Although they were based on a small population, the resulting burden rates were comparable to those experienced, in general, by other EMD programs. Appendix F contains the resulting average burden rates. These rates were applied to the estimated software cost estimate to arrive at a software price regression. Programs NCCA 4 and NCCA 15 provided cost through G&A. Typically COM is applied to cost less G&A, versus COM being applied to cost through G&A, but in order to remain consistent, NCCA calculated and then applied an average COM rate, 2.1 percent to cost through G&A. The average COM rate was based on six programs from the NCCA Normalized Labor Rate Database. An average fee, 10.9 percent, calculated based on three programs that provided fee separately, was then applied to cost

Section 9 – Labor Rate Analysis

through G&A and COM. The following regression was developed to estimate the price of software through G&A, COM and fee:

$$\text{FY97\$K} = 154.21 * (\text{Labor Khrs})^{0.98} * e^{(D_1 * -0.39)}$$

$R^2 = 0.99$; $CV = 0.13$; **Predict (20) = 87%**; **Range 2.41 - 792.80 Labor Khours**
where $D_1 = 0$ for aircraft and 1 for non-aircraft

[9-5a]

NCCA's recommended process for converting effort-to-cost is shown in Figure 9-5.

Step 1: Determine if there is additional data (analogous or contractor). If so, then normalize the data and develop analogous or contractor specific cost estimating tools.

Step 2: If no data exists, then use NCCA's effort-to-cost conversion regression (aircraft and non-aircraft), provided above.

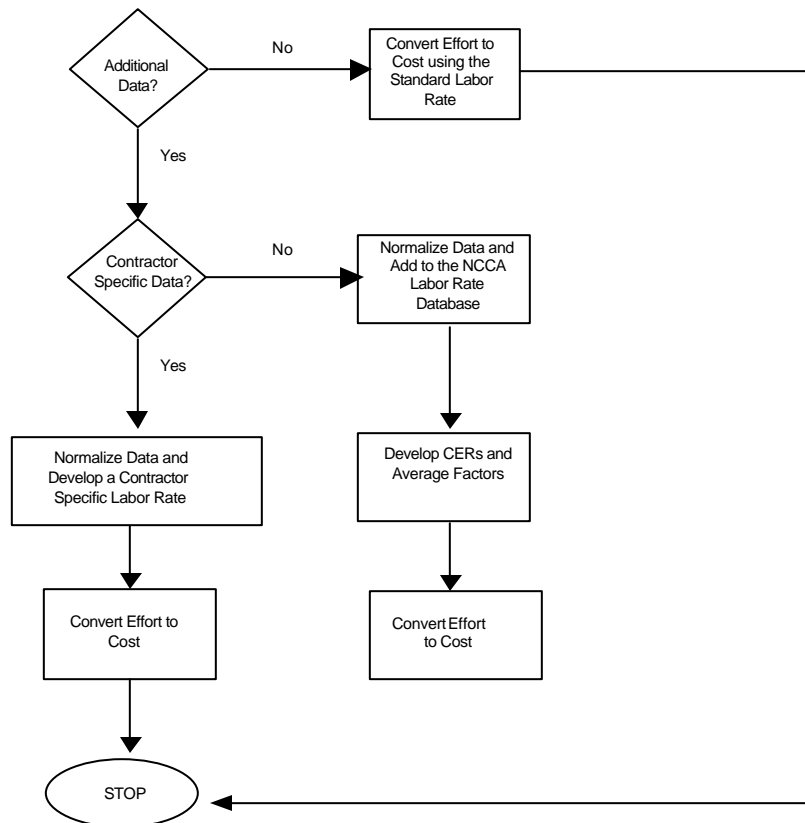


Figure 9-5: Recommended Labor Rate Estimation Process

9.5 CONCLUSIONS

With the exception of conducting risk analysis, converting effort-to-cost is the final step towards developing a software cost estimate. Through this analysis, NCCA was able to identify platform type as a variable that affects the cost of software development. This variable was the only significant variable statistically proven to affect software development cost. Even with the exceptional statistics and robust underlying data set (i.e., size, range, platform type, contract type, etc.), this analysis could be enhanced with the collection of more data; especially data that reflects current practices and technologies.

9.6 ADDITIONAL CONSIDERATIONS

There are two other issues the analyst should consider when utilizing the NCCA Raw Labor Rate Database or historical data from other sources. These issues are: 1) drastic overhead rate changes due to company location or business base changes and 2) acquisition strategy changes.

If a company relocates, the relocation can affect the labor hours expended (both indirect and direct costs). If a company has already relocated, NCCA recommends obtaining the projected change in rates from the Administrative Contracting Officer (ACO). Additionally, different divisions within a company may have significantly different productivities and rates due to different business practices and/or accounting structures. If a company consolidates with other divisions or companies, drastic changes in rate structures may occur.

Acquisition strategy also affects the labor cost. An analyst may see exceptionally low direct labor and overhead rates if the contractor is using software vendor houses vice developing software in-house. The NCCA analyst needs to know who is developing the software and where, and also, how this compares with the way business was previously conducted.

If the analyst determines, that a contractor's proposed labor rate is relatively low in comparison to historical labor rates for that contractor or NCCA's average labor rates, the analyst should investigate whether or not there have been any changes to the company's rate structure. If there have been changes, determine the underlying cause of these changes. If the contractor does not provide valid reasons for the changes, the analyst should use NCCA's recommended approach.

9.7 FUTURE EFFORTS

In the future, to improve the effort-to-cost conversion tool, NCCA recommends the following efforts be performed:

- 1) Collect data for additional embedded programs to enlarge the NCCA Normalized Labor Rate Database.
- 2) Collect data for MIS programs to widen the software application range of the NCCA Normalized Labor Rate Database.

Section 9 – Labor Rate Analysis

- 3) Collect SLOC and cost per reporting period for each program in order to evaluate the relationship between SLOC growth (an indication of problems in the development process) and the associated software cost. This data will also allow us to track the average man-year rate and software team composition over the software development process. Appendix F provides more details.
- 4) Collect both CCDRs and CPRs for programs to investigate the differences between percent expended and percent complete.
- 5) Track the software development effort and associated man-year rates for specific contractors. Collecting effort data in conjunction with the associated man-year rates would provide insight into the specific skill level that is required to achieve a certain productivity level. For example, this would allow the analyst to obtain the associated man-year rate of z_1 for program X which experienced a productivity of seven SLOC per day versus program Y which experienced four SLOC per day with an associated man-year rate of z_2 .
- 6) Collect the phases associated with the hours expended developing the software.
- 7) Collect the actual hours expended per man-month for data collected via CPRs which report total man-months expended for software development.

RISK ANALYSIS

10.1 INTRODUCTION

Since software size is typically one of the primary inputs for developing a software cost estimate, the accuracy of the cost estimate is highly dependent on the quality of the size estimate. However, size is difficult to estimate early in the acquisition cycle when the system design and requirements are not clearly defined. Verner and Tate [20] cited several studies that pursued the “elusive goal” of accurate size estimation, and Conte [21] stated that “expert sizing depends on so many subjective factors that different ‘experts’ often arrive at radically different estimates.”

Historically, SLOC size estimates have been optimistically low with respect to total code count and optimistically high with respect to reused code count. This section of the handbook addresses NCCA’s approach to quantifying the risk associated with optimistic size and reuse assumptions.

The following areas of discussion describe NCCA’s development of a risk analysis methodology:

- NCCA Risk Analysis Databases
- SLOC Growth Methodology, Results and Conclusions
- Code Condition Change Methodology, Results and Conclusions
- Overall Recommended Approach
- Future Efforts

10.2 NCCA RISK ANALYSIS DATABASES

NCCA also created a separate SLOC growth risk database to support the development of software risk analyses. The detailed methodology will be discussed below.

10.2.1 GROUND RULES AND ASSUMPTIONS

Listed below are the ground rules and assumptions for this analysis:

- 1) NCCA assumed that the program-level sample data was derived from a normally-distributed population. This means that calculating the mean, median, and mode statistics is simple, and the procedure is well documented. Additionally, the results of statistical tests, such as the t-test, are valid for making decisions about different samples.
- 2) All program-level SLOC represented logical lines of code.

10.2.2 RAW RISK ANALYSIS DATABASE

NCCA extracted the majority of the data from the SMC Database [7] and the remaining data from internal NCCA files and a study by Om and Bui [22] at IDA.

NCCA performed a query of the SMC Database to obtain data points that included both actual and estimated SLOC measured using the logical code counting convention. The query produced 12 program-level and 28 CSCI-level data points. Next, a search of NCCA's files provided 11 additional program-level data points. Finally, one program-level and four CSCI-level data points were collected from the IDA study for a total of 23 program-level and 32 CSCI-level data points. See Appendix G for the list of data points, including associated data elements.

10.2.3 NCCA NORMALIZED RISK ANALYSIS DATABASE

A number of data points in the raw database were eliminated. Specifically, three of the 23 program-level data points were MIS programs which were excluded to remain consistent with the effort, schedule and labor rate analyses. All 32 of the CSCI-level data points were excluded based on the results of the Mann Whitney U test and the Kolmogorov-Smirnov test (see Appendix C for more details), which showed that the means and variances were not equal to the program-level data points. In addition to these deletions, four program-level data points that did not have verifiable SLOC estimates were also eliminated. Finally, the Kolmogorov-Smirnov test was conducted to determine whether the one Assembly program should be deleted. It was determined that the one Assembly data point was not statistically different than the other 15 HOL data points. As a result, the normalized SLOC Growth Database includes 16 program-level data points.

Although the programs were developed between Milestones II and III, specific review dates were unknown. The majority of the program names are also unknown. The range of estimated SLOC values is 14 to 1,246 KSLOC; nine programs are less than 100 KSLOC. Five programs are entirely new. All the data is for weapon system programs where the condition of the code, both new and reused, was known. See Appendix G for a detailed list of these data points with associated data elements.

The program-level data was partitioned in several ways: 1) language (Fortran, Ada, Jovial, Atlas, CMS-2, C or C++, Assembly, and other); 2) development method (waterfall, incremental, and spiral); 3) mission assigned (Command and Control (C²), testing, software tools, signal processing, and mission plans); and 4) complexity (simple, routine, difficult and complex). Information on the development method and complexity were only available for the SMC data points. All variables cited above were objective measures except complexity level which was a subjective measure of requirements definition complexity:

- 1) Simple - Existing product line in an existing environment.
- 2) Routine - New product line in an existing environment.
- 3) Difficult - New product line in a new environment.
- 4) Complex - Pushing state-of-the-art.

10.3 SLOC GROWTH METHODOLOGY, RESULTS AND CONCLUSIONS

Three different SLOC growth estimating approaches were developed. Each approach has two sections: 1) Methodology and Results and 2) Conclusions.

10.3.1 APPROACH ONE

10.3.1.1 APPROACH ONE METHODOLOGY AND RESULTS

The first approach established top-level standard growth factors by calculating the mean, median, and range mode of the percent total growth, where:

$$\% \text{Total Growth} = \frac{(\text{Actual SLOC} - \text{Estimated SLOC})}{\text{Estimated SLOC}} * 100$$

The median and range mode statistics indicated the “most likely” percent growth. The standard factor, when applied as shown below, provides a revised SLOC estimate. (See examples on pages 10-16 through 10-19.)

$$\text{Revised SLOC} = (\% \text{Total Growth} * \text{Initial SLOC Estimate}) + \text{Initial SLOC Estimate}$$

The mean, SEE, and CV_{est} were calculated for the percent total growth. See Table 10-1. The Predict (20) was also calculated for each metric. (See Appendix C for a more complete explanation of CV_{est} and Predict (20).)

As shown in Table 10-1, the mean percent total growth for the 16 programs was 63 percent or 59 KSLOC. The average actual size of a program was 274 KSLOC, and the Predict (20) was 25 percent; that is, four of the 16 estimated values were within 20 percent of their actual percent growth. The CV_{est} for the mean was 82 percent; the mean overestimated 69 percent of the programs possibly due to four large-growth data points (#18, #327, #2461, and NCCA-1). Thus, it was not a good predictor of total growth.

Since the mean overestimated a disproportionate amount of the time, NCCA performed a tradeoff analysis of the data to determine whether program size influenced the percentage of growth experienced. This tradeoff analysis (similar to the one discussed previously for E factors in Section 5 - **Effort Analysis: Normalized Regressions**) determined the point where the lowest CV occurred. This “optimal” point was 100 KSLOC with nine programs less than 100 KSLOC and seven programs greater than 100 KSLOC. However, the associated t-statistic for this variable was not significant at the 95 percentile. Furthermore, when these samples were compared using the Mann-Whitney U test, the results demonstrated that the means of the two samples were equal (see Appendix G for the data and non-parametric results); hence, no further analysis of the separate samples was done.

Section 10 – Risk Analysis

| | | A | B | | | C | D | |
|-------------|---------------|---------|---------|--------------|-------------|---------------------|------------|---------|
| Record # | Mission | Est Tot | Act Tot | Total Growth | %Tot Growth | Y(est) | e(i) | %e(i) |
| | | | | (B - A) | (B/A) - 1 | (0.6326 x A) + A | (C - B) | (D/B) |
| 8 | C2 | 618000 | 709000 | 91000 | 14.72 | 621909.31 | -87090.69 | -12.28% |
| 15/16/17 | C2 | 23599 | 25814 | 2215 | 9.39 | 23748.28 | -2065.72 | -8.00% |
| 18 | C2 | 14000 | 70143 | 56143 | 401.02 | 14088.56 | -56054.44 | -79.91% |
| 23/24/26/27 | Testing | 41800 | 46303 | 4503 | 10.77 | 42064.42 | -4238.58 | -9.15% |
| 308 | S/W Tools | 45000 | 45000 | 0 | 0.00 | 45284.66 | 284.66 | 0.63% |
| 327 | C2 | 39294 | 119400 | 80106 | 203.86 | 39542.56 | -79857.44 | -66.88% |
| 2459 | C2 | 22000 | 30000 | 8000 | 36.36 | 22139.17 | -7860.83 | -26.20% |
| 2461 | Signal Proc | 15500 | 26513 | 11013 | 71.05 | 15598.05 | -10914.95 | -41.17% |
| 2613 | C2 | 100000 | 122000 | 22000 | 22.00 | 100632.57 | -21367.43 | -17.51% |
| 2616 | Mission Plans | 532000 | 877129 | 345129 | 64.87 | 535365.30 | -341763.70 | -38.96% |
| NCCA-1 | C2 | 206650 | 394309 | 187659 | 90.81 | 207957.22 | -186351.78 | -47.26% |
| NCCA-2 | C2 | 74000 | 82930 | 8930 | 12.07 | 74468.11 | -8461.89 | -10.20% |
| NCCA-3 | C2 | 213800 | 261800 | 48000 | 22.45 | 215152.44 | -46647.56 | -17.82% |
| NCCA-4 | C2 | 153000 | 185000 | 32000 | 20.92 | 153967.84 | -31032.16 | -16.77% |
| NCCA-5 | C2 | 83900 | 108850 | 24950 | 29.74 | 84430.73 | -24419.27 | -22.43% |
| NCCA-6 | C2 | 1246272 | 1272200 | 25928 | 2.08 | 1254155.60 | -18044.40 | -1.42% |
| | | | | | | | | |
| N = 16 | | Mean = | 273524 | 59224 | 63.26% | SEE = | 111299 | |
| | | | | | | | | |
| | | | | | | CV _{est} = | 0.41 | |
| | | | | | | | | |
| | | | | | | Predict (20) = | 25% | |

Table 10-1: Mean Percent SLOC Growth Analysis

To minimize the impact of the large-growth data points, NCCA developed an alternative growth factor based on the median (vice mean) of the NCCA Normalized SLOC Growth Database. The median percent total growth was 22 percent or 35 KSLOC. By definition, the median splits the data into two equal parts; hence, the median percent total growth neither overestimated nor under-estimated a disproportionate amount of the time, and the CV_{est} decreased substantially. The CV_{est} when applying the median was 37 percent and the Predict (20) was 62.5 percent. Therefore, even though a program experienced a very small or very large amount of growth, the median was within 20 percent of the actual values 63 percent of the time. Hence, it performed better than the mean (see Table 10-2).

In addition to evaluating the mean- and median-based growth factors, the range mode was considered. The range mode percent total growth was 20 to 30 percent. This metric provided a range estimate rather than a single point estimate.

10.3.1.2 APPROACH ONE CONCLUSIONS

The advantages of these top level factors are: 1) the mean and median percent total growth are simple to calculate and apply; 2) the median percent total growth is relatively insensitive to extreme values; and 3) the median percent total growth factor falls within the range mode and therefore is assumed to be more accurate than the mean.

Section 10 – Risk Analysis

| | | A | B | | | C | D | |
|-------------|---------------|---------|---------|--------------|-------------|---------------------|------------|---------|
| Record # | Mission | Est Tot | Act Tot | Total Growth | %Tot Growth | Y(est) | e(i) | %e(i) |
| | | | | (B - A) | (B/A) - 1 | (0.2225 x A) + A | (C - B) | (D/B) |
| 308 | S/W Tools | 45000 | 45000 | 0 | 0.00 | 55012.50 | 10012.50 | 22.25% |
| NCCA-6 | C2 | 1246272 | 1272200 | 25928 | 2.08 | 1523567.52 | 251367.52 | 19.76% |
| 15/16/17 | C2 | 23599 | 25814 | 2215 | 9.39 | 28849.78 | 3035.78 | 11.76% |
| 23/24/26/27 | Testing | 41800 | 46303 | 4503 | 10.77 | 51100.50 | 4797.50 | 10.36% |
| NCCA-2 | C2 | 74000 | 82930 | 8930 | 12.07 | 90465.00 | 7535.00 | 9.09% |
| 8 | C2 | 618000 | 709000 | 91000 | 14.72 | 755505.00 | 46505.00 | 6.56% |
| NCCA-4 | C2 | 153000 | 185000 | 32000 | 20.92 | 187042.50 | 2042.50 | 1.10% |
| 2613 | C2 | 100000 | 122000 | 22000 | 22.00 | 122250.00 | 250.00 | 0.20% |
| NCCA-3 | C2 | 213800 | 261800 | 48000 | 22.45 | 261370.50 | -429.50 | -0.16% |
| NCCA-5 | C2 | 83900 | 108850 | 24950 | 29.74 | 102567.75 | -6282.25 | -5.77% |
| 2459 | C2 | 22000 | 30000 | 8000 | 36.36 | 26895.00 | -3105.00 | -10.35% |
| 2616 | Mission Plans | 532000 | 877129 | 345129 | 64.87 | 650370.00 | -226759.00 | -25.85% |
| 2461 | Signal Proc | 15500 | 26513 | 11013 | 71.05 | 18948.75 | -7564.25 | -28.53% |
| NCCA-1 | C2 | 206650 | 394309 | 187659 | 90.81 | 252629.63 | -141679.38 | -35.93% |
| 327 | C2 | 39294 | 119400 | 80106 | 203.86 | 48036.92 | -71363.09 | -59.77% |
| 18 | C2 | 14000 | 70143 | 56143 | 401.02 | 17115.00 | -53028.00 | -75.60% |
| | | | | | | | | |
| N = 16 | | Mean = | 273524 | Median = | 22.25% | SEE = | 101786 | |
| | | | | | | | | |
| | | | | | | CV _{est} = | 0.37 | |
| | | | | | | | | |
| | | | | | | Predict (20) = | 62.50% | |

Table 10-2: Median Percent SLOC Growth Analysis

Generally, the disadvantages of this approach are: 1) the mean is much more sensitive to extreme values (as demonstrated by the large CV_{est} and small Predict (20)); 2) the median ignores the relative size of the apparent growth, treating all data points greater than or less than the median point equally (i.e., all the data points with greater than 22 percent growth are treated identically); and 3) a point regression may not provide an accurate “most likely” estimate.

10.3.2 APPROACH TWO

10.3.2.1 APPROACH TWO METHODOLOGY AND RESULTS

The second approach used regression analysis to develop relationships of the form:

$$\begin{aligned}\text{Actual SLOC} &= f(\text{Estimated SLOC}) \\ \text{Percent Growth} &= f(\text{Estimated SLOC})\end{aligned}$$

These regressions were developed with and without dummy variables, such as: size, language, development method, complexity, mission, and percent actual new code. See Appendix C for an explanation of dummy variables.

The validity and significance of the resulting regression equations were evaluated using a variety of statistical tests and measures such as the F- and t-tests, Mann-Whitney U test, Kolmogorov-Smirnov test, CV, R^2 values, SEE, and residual analysis.

Section 10 – Risk Analysis

Table 10-3 displays representative regression equations and their statistics:

| Y(est) | = | X ₁ (coeff) | X ₁ | + | X ₂ (coeff) | X ₂ | + | Constant | R ² | CV _{est} | Std ERR | Predict (20) | t ₁ (value) | t ₁ (sig) | t ₂ (value) | t ₂ (sig) | N | df | F(value) | F(sig) |
|-----------------|---|------------------------|----------------|---|------------------------|----------------|---|-----------|----------------|-------------------|----------|--------------|------------------------|----------------------|------------------------|----------------------|----|----|----------|--------|
| Act Tot | = | 1.0816 | Est Tot | + | | | + | 41743 | 0.95 | 0.33 | 88901.92 | 31% | 15.506 | 100% | | | 16 | 14 | 240.44 | 100% |
| ln(Act Tot) | = | 0.00000315 | Est Tot | + | | | + | 11.0942 | 0.68 | 0.74 | 0.7368 | 13% | 5.4426 | 100% | | | 16 | 14 | 29.62 | 100% |
| ln(Act Tot) | = | 0.8769 | ln(Est Tot) | + | | | + | 1.7791 | 0.90 | 0.42 | 0.4163 | 44% | 11.0725 | 100% | | | 16 | 14 | 122.60 | 100% |
| %Tot Growth | = | -7.51876E-07 | Est Tot | + | | | + | 0.7937 | 0.06 | 1.64 | 1.0381 | 6% | -0.9231 | 63% | | | 16 | 14 | 0.85 | 63% |
| ln(%Tot Growth) | = | -0.0000021 | Est Tot | + | | | + | -0.6690 | 0.29 | 1.14 | 1.1351 | 6% | -2.41 | 97% | | | 16 | 14 | 5.81 | 97% |
| ln(%Tot Growth) | = | -0.4727 | ln(Est Tot) | + | | | + | 4.2560 | 0.24 | 1.18 | 1.1756 | 31% | -2.114 | 95% | | | 16 | 14 | 4.47 | 95% |
| Act Tot | = | 0.959 | Est Tot | + | 118245.74 | Size > 100K | + | 23668.443 | 0.96 | 0.29 | 79722.18 | 38% | 11.2093 | 100% | 2.0999 | 94% | 16 | 13 | 151.70 | 100% |
| Act Tot | = | 0.9981 | Est Tot | - | 86381.26 | Size < 100K | + | 108219.09 | 0.95 | 0.31 | 84391.81 | 25% | 11.8198 | 100% | -1.5926 | 86% | 16 | 13 | 134.68 | 100% |
| Act Tot | = | 1.0756 | Est Tot | - | 31360.48 | Ada | + | 48893.809 | 0.95 | 0.33 | 91277.09 | 38% | 14.8398 | 100% | -0.53 | 39% | 16 | 13 | 114.18 | 100% |
| Act Tot | = | 1.0881 | Est Tot | - | 47798.32 | C ² | + | 76186.421 | 0.95 | 0.33 | 89384.89 | 38% | 15.4364 | 100% | -0.9215 | 63% | 16 | 13 | 119.35 | 100% |
| Act Tot | = | 1.3715 | Est Tot | - | 80761.80 | Routine | + | 32333.65 | 0.95 | 0.34 | 69891 | 20% | 13.11 | 100% | -1.63 | 85% | 10 | 7 | 86.90 | 100% |
| Act Tot | = | 1.3504 | Est Tot | - | 72690.60 | Waterfall | + | 40243.39 | 0.95 | 0.34 | 70347 | 30% | 13.10 | 100% | -1.59 | 84% | 10 | 7 | 85.80 | 100% |

Table 10-3: Summary of Approach Two - Size Growth Estimating Relationships

The R² values demonstrate that this approach produced valid regressions for estimating actual total SLOC (the first three equations in Table 10-3 above), but not for estimating percent growth (the middle three equations). Appendix G contains the regression analyses for the significant regressions (first six equations above). Since none of the dummy variables attempted in equations six through 12 (the last six equations) were significant at the 95 percent confidence level, they were eliminated from further consideration. Of the first six equations, the growth estimating relationship with the lowest CV and best associated statistics follows:

$$\text{Actual Total SLOC} = 41743 + (1.0816 * \text{Estimated Total SLOC}) \quad [10-1]$$

$$R^2 = 0.95; CV_{\text{est}} = 0.33; \text{Predict (20)} = 31\%; \text{Range} = 14 - 1,246 \text{ KSLOC}$$

However, as shown by the residuals in Table 10-4, the equation severely overestimated most of the smaller programs.

10.3.2.2 APPROACH TWO CONCLUSIONS

The use of regression analysis indicates that there is a strong relationship between actual total SLOC and estimated total SLOC. The advantage to Approach Two is that a statistical model explains the systematic behavior of the data while leaving out random components.

The disadvantages to Approach Two are: 1) the influence of an excessively large program (e.g. NCCA-6) with a small growth percentage on the slope of the regression line (i.e., disproportionate amounts of over- or underestimating); and 2) the similar effect of an excessively small program (like #18) with a large growth percentage on the slope of the regression line.

| Record # | Mission | Est Tot | Act Tot | A | B | C |
|-------------|---------------|---------|---------|------------|------------|---------|
| | | | | Y(est) | e(i) | %e(i) |
| | | | | | (B - A) | (C/A) |
| 15/16/17 | C2 | 23599 | 25814 | 67267.16 | 41453.16 | 160.58% |
| 2461 | Signal Proc | 15500 | 26513 | 58507.53 | 31994.53 | 120.67% |
| 2459 | C2 | 22000 | 30000 | 65537.73 | 35537.73 | 118.46% |
| 308 | S/W Tools | 45000 | 45000 | 90413.81 | 45413.81 | 100.92% |
| 23/24/26/27 | Testing | 41800 | 46303 | 86952.79 | 40649.79 | 87.79% |
| 18 | C2 | 14000 | 70143 | 56885.18 | -13257.82 | -18.90% |
| NCCA-2 | C2 | 74000 | 82930 | 121779.31 | 38849.31 | 46.85% |
| NCCA-5 | C2 | 83900 | 108850 | 132486.84 | 23636.84 | 21.72% |
| 327 | C2 | 39294 | 119400 | 84242.38 | -35157.62 | -29.45% |
| 2613 | C2 | 100000 | 122000 | 149900.10 | 27900.10 | 22.87% |
| NCCA-4 | C2 | 153000 | 185000 | 207223.25 | 22223.25 | 12.01% |
| NCCA-3 | C2 | 213800 | 261800 | 272982.64 | 11182.64 | 4.27% |
| NCCA-1 | C2 | 206650 | 394309 | 265249.42 | -129059.58 | -32.73% |
| 8 | C2 | 618000 | 709000 | 710152.78 | 1152.78 | 0.16% |
| 2616 | Mission Plans | 532000 | 877129 | 617137.86 | -259991.14 | -29.64% |
| NCCA-6 | C2 | 1246272 | 1272200 | 1389672.23 | 117472.23 | 9.23% |

Table 10-4: Residuals of Actual Total SLOC versus Estimated Total SLOC

10.3.3 APPROACH THREE

10.3.3.1 APPROACH THREE METHODOLOGY AND RESULTS

The third approach was a two-step analysis based on the hypothesis that size growth is inversely proportional to the extent of code reused (i.e., greater reuse means less growth and vice versa).

The first step was to determine the program's percentage of reused SLOC. This was accomplished by estimating percent actual new SLOC (1 - percent actual reused SLOC) based on percent estimated new SLOC, since percent actual new SLOC (or percent actual reused SLOC) is unknown when a program initially starts. The form of the regression is as follows:

$$\% \text{Actual New SLOC} = a + (b * \% \text{Estimated New SLOC})$$

The second step was to estimate the actual SLOC based on the estimated SLOC and a dummy variable which accounts for the percentage of reused SLOC (1 - percent estimated new SLOC) in the program. The dummy variable (percent actual new SLOC) was based on the estimating methodology described in the first step. Below is the linear equation for the second step:

$$\text{Actual SLOC} = a + (b * \text{Estimated SLOC}) + (c * \% \text{Actual New SLOC (estimated)})$$

Section 10 – Risk Analysis

Table 10-5 below shows the percent actual and estimated new SLOC and percent actual and estimated reused SLOC that were used in performing the regression analysis. Only 11 of the 16 data points were included, since five were entirely new programs.

| Record # | Mission | %New Est | %New Act | %Reuse Est | %Reuse Act |
|-------------|---------------|----------|----------|------------|------------|
| 15/16/17 | C2 | 11.71 | 19.56 | 88.29 | 80.44 |
| 23/24/26/27 | Testing | 88.28 | 100.00 | 11.72 | 0.00 |
| 308 | S/W Tools | 44.44 | 55.56 | 55.56 | 44.44 |
| 327 | C2 | 75.14 | 100.00 | 24.86 | 0.00 |
| 2461 | Signal Proc | 95.48 | 96.38 | 4.52 | 3.62 |
| 2613 | C2 | 90.00 | 98.36 | 10.00 | 1.64 |
| 2616 | Mission Plans | 24.81 | 61.24 | 75.19 | 38.76 |
| NCCA-2 | C2 | 25.68 | 33.68 | 74.32 | 66.32 |
| NCCA-3 | C2 | 89.50 | 80.42 | 10.50 | 19.58 |
| NCCA-4 | C2 | 42.48 | 86.49 | 57.52 | 13.51 |
| NCCA-6 | C2 | 41.49 | 40.84 | 58.51 | 59.16 |
| N = 11 | Mean = | 57.18 | 70.23 | 42.82 | 29.77 |

Table 10-5: Programs with Reuse

Table 10-6 shows the results of all three forms of the regression equation (see Appendix G). The R^2 values demonstrated that this approach produced adequate regressions for percent actual new SLOC for the linear and power forms of the regressions.

| Y(est) | = | X_1 (coeff) | X_1 | + | Constant | R^2 | CV_{est} | Std ERR | Predict (20) | t(value) | t(sig) | N | df | F(value) | F(sig) |
|--------------|---|---------------|--------------|---|----------|--------|------------|---------|--------------|----------|--------|----|----|----------|--------|
| %Act New | = | 0.8189 | %Est New | + | 0.234 | 0.7426 | 0.22 | 0.1575 | 45% | 5.0957 | 99.94% | 11 | 9 | 25.9665 | 100% |
| ln(%Act New) | = | 1.4469 | %Est New | + | -1.2919 | 0.6863 | 0.3196 | 0.3196 | 55% | 4.4376 | 99.84% | 11 | 9 | 19.6919 | 100% |
| ln(%Act New) | = | 0.7057 | ln(%Est New) | + | 0.0563 | 0.7965 | 0.2575 | 0.2575 | 55% | 5.9347 | 99.90% | 11 | 9 | 35.2207 | 100% |

Table 10-6: Approach Three Size Growth Estimating Relationships

The best regression equation and its associated statistics follow:

$$\begin{aligned} \text{\%Actual New SLOC} &= 0.234 + (0.8189 * \text{\%Estimated New SLOC}) \\ R^2 &= 0.74; CV = 0.22; \text{Predict (20)} = 45\%; \text{Range} = 16 - 1,246 \text{ KSLOC} \end{aligned} \quad [10-2]$$

Although NCCA was able to develop a statistically significant percent new SLOC regression, this approach was abandoned because percent actual new was insignificant (i. e., percent reused SLOC did not drive SLOC growth) as a dummy variable (see $t_2(\text{sig})$ in Table 10-7). The following equation represents the form of the regression:

$$\begin{aligned} \text{Actual SLOC} &= a + (b * \text{Estimated SLOC}) + (c * \text{\%Actual New SLOC}) \\ b &= x_1(\text{coefficient}); c = x_2(\text{coefficient}) \end{aligned}$$

Table 10-7 lists representative regressions and their statistics (see Appendix G):

Section 10 – Risk Analysis

| Y(est) | = | X ₁ (coeff) | X ₁ | + | X ₂ (coeff) | X ₂ | + | Constant | R ² | CV | Std ERR | Predict (20) | t ₁ (value) | t ₁ (sig) | t ₂ (value) | t ₂ (sig) | N | df | F(value) | F(sig) |
|-----------------|---|------------------------|----------------|---|------------------------|----------------|---|----------|----------------|--------|---------|--------------|------------------------|----------------------|------------------------|----------------------|----|----|----------|--------|
| Act Tot | = | 1.0878 | Est Tot | + | 37609.89 | %Act New | + | 6462.427 | 0.9462 | 0.38 | 106225 | 27% | 11.3081 | 100.00% | 0.3117 | 23.67% | 11 | 8 | 70.3836 | 100% |
| ln(Act Tot) | = | 3.18E-06 | Est Tot | + | 0.9134 | %Act New | + | 10.3568 | 0.733 | 0.7525 | 0.7525 | 9% | 4.6614 | 99.84% | 1.0684 | 68.35% | 11 | 8 | 10.9819 | 99% |
| ln(Act Tot) | = | 0.9708 | ln(Est Tot) | + | 0.5154 | %Act New | + | 0.2505 | 0.953 | 0.3157 | 0.3157 | 55% | 12.6866 | 100.00% | 1.4884 | 82.50% | 11 | 8 | 81.144 | 100% |
| %Tot Growth | = | -7.3E-08 | Est Tot | + | 0.8772 | %Act New | + | -0.2001 | 0.2057 | 1.48 | 0.5898 | 9% | -0.1368 | 10.54% | 1.3092 | 77.32% | 11 | 8 | 1.0362 | 60% |
| ln(%Tot Growth) | = | -1.54E-06 | Est Tot | + | 1.4108 | %Act New | + | -2.025 | 0.3901 | 1.1306 | 1.1306 | 0% | -1.5087 | 83.02% | 1.0984 | 69.60% | 11 | 8 | 2.5587 | 86% |
| ln(%Tot Growth) | = | -0.342 | ln(Est Tot) | + | 1.7259 | %Act New | + | 1.3164 | 0.3353 | 1.1803 | 1.1803 | 27% | -1.1954 | 73.38% | 1.3329 | 78.07% | 11 | 8 | 2.0181 | 80% |

Table 10-7: Software Growth Estimating Relationships

10.3.3.2 APPROACH THREE CONCLUSIONS

The advantages of the third approach are similar to those cited for Approach Two. The primary disadvantage is that the dummy variable (percent actual new SLOC) is not statistically significant, so further consideration of this approach was ceased. Also, this approach adds an extra layer of uncertainty when estimating future programs due to the addition of a second regression.

10.3.4 RECOMMENDED APPROACH

Table 10-8 summarizes the statistics for: 1) the two factors developed in Approach One and 2) the most statistically significant regression equation developed in Approach Two.

| | Estimating Methodology | CV _{est} | Predict (20) |
|---------------------|---|-------------------|--------------|
| Mean | 63% | 0.82 | 25.00% |
| Median | 22% | 0.37 | 62.50% |
| Regression Equation | Actual Total SLOC = 41,743 + (1.081 * Est Total SLOC) | 0.33 | 31.00% |

Table 10-8: Summary of the Statistics for SLOC Growth Methodology

In Approach One, the application of the mean resulted in a high CV and low Predict (20) value. The mean also overestimated a significant amount of the time. The median, however, provided the highest Predict (20) and one of the lowest CVs for the resulting estimate, and, by definition, neither over- nor underestimated disproportionately.

In Approach Two, the results of the simple regression performed on total SLOC showed an extremely high R² value, indicating a good regression and a relatively low CV; however, the Predict (20) was low and the equation tended to overestimate. Approach Three results were not statistically significant.

To minimize the potential of overestimating a disproportionate amount of the time, NCCA prefers to maximize Predict (20) (i.e., be closer to the most likely estimate a higher percentage of the time) rather than minimize overall error. (As stated previously, this is in contrast to our philosophy of minimizing standard error that was applied in earlier sections of this handbook.) Therefore, based on the results of Approaches One and Two (since Approach Three was not significant), NCCA recommends applying the median percent total growth factor of 22 percent when contractor/program-specific data is unavailable for developing specific growth estimating regressions or factors in a particular program. The major strength of this approach is that this factor represents the “most likely” growth.

10.4 CODE CONDITION CHANGE METHODOLOGY, RESULTS AND CONCLUSIONS

Not only does the SLOC count change over time, but usually the code condition also changes. The code condition (type and amount of reused and new SLOC) is an important input for generating an ESLOC estimate and subsequent effort estimate. Initial estimates generally overestimate the amount of reused SLOC and correspondingly underestimate the amount of new SLOC (as shown in Table 10-5 previously). Since reused code is typically less costly to develop than new code, a change in the reused versus new code distribution can cause a significant change in cost. In addressing these issues, NCCA used the approaches described in previous sections to develop the means of overcoming flawed code condition estimating practices.

This section describes the procedures for adjusting the distribution of the estimated reused and estimated new SLOC counts using the top-level standard factor approach (Approach One) and regression analysis approach (Approach Two) as described previously.

As discussed in Section 4 - **Effort Analysis: Significant Drivers**, ESLOC are the weighted sum of new and reused SLOC. Generally, the effort associated with developing reused code is less than the effort to develop new lines of code, since reused code does not go through the full software development process. Therefore, to accurately estimate the effort associated with a program, the differences between new and reused SLOC must be considered. This is accomplished by calculating ESLOC as follows:

$$\text{Equivalent SLOC} = \text{New SLOC} + (\text{Efactor} * \text{Reused SLOC})$$

where Efactor (as discussed in Sections 4 through 6) is calculated quantitatively.

10.4.1 APPROACH ONE

Four standard factors were developed for the first approach: mean percentage, median percentage, mean percentage points, and median percentage points. Means, SEEs, CVs, Predict (20)s, and residuals were calculated in each case.

10.4.1.1 MEAN PERCENTAGE

The Mean Percentage is the average percentage increase in the estimated percent new SLOC as follows:

$$\text{Mean Percentage Increase} = \frac{\sum \frac{\% \text{New Actual SLOC}}{\% \text{Estimated New SLOC}} - 1}{n} * 100$$

This percentage was then applied to the estimated percent new SLOC. The disadvantage of this factor was that the Mean Percentage was sensitive to extreme values.

Section 10 – Risk Analysis

The Mean Percentage Increase was 38 percent with a Predict (20) of 36 percent⁶² and a CV_{est} of 37 percent. The residuals in Table 10-9 indicated that Mean Percentage Increase overestimated 73 percent of the time.

| Record # | Mission | Est New | Actual New | A | B | C | D |
|-------------|---------------|---------|------------|----------|----------|-------------|---------------------|
| | | | | %New Est | %New Act | %New Growth | Y(est) |
| | | | | | | (B/A) - 1 | (0.38 x A) + A |
| 15/16/17 | C2 | 2763 | 5048 | 11.71 | 19.56 | 67.02 | 16.16 |
| 23/24/26/27 | Testing | 36900 | 46303 | 88.28 | 100.00 | 13.28 | 121.86 |
| 308 | S/W Tools | 20000 | 25000 | 44.44 | 55.56 | 25.00 | 61.35 |
| 327 | C2 | 29524 | 119400 | 75.14 | 100.00 | 33.09 | 103.72 |
| 2461 | Signal Proc | 14800 | 25552 | 95.48 | 96.38 | 0.93 | 131.81 |
| 2613 | C2 | 90000 | 120000 | 90.00 | 98.36 | 9.29 | 124.24 |
| 2616 | Mission Plans | 132000 | 537129 | 24.81 | 61.24 | 146.80 | 34.25 |
| NCCA-2 | C2 | 206650 | 394309 | 25.68 | 33.68 | 31.17 | 35.44 |
| NCCA-3 | C2 | 191350 | 210550 | 89.50 | 80.42 | -10.14 | 123.55 |
| NCCA-4 | C2 | 65000 | 160000 | 42.48 | 86.49 | 103.58 | 58.65 |
| NCCA-6 | C2 | 517071 | 519600 | 41.49 | 40.84 | -1.56 | 57.27 |
| N = 11 | | | Mean = | 57.18 | 70.23 | 38.04% | SEE = |
| | | | | | | | CV _{est} = |
| | | | | | | | Predict (20) = |

Table 10-9: Code Condition Mean Percentage Statistics

10.4.1.2 MEDIAN PERCENTAGE

The median percentage used the “most likely” increase in percent estimated new SLOC. It was the central value that divided the data into two groups of equal size and was calculated from the percent actual increase of each program. The advantage of this factor was that the median percentage was relatively insensitive to extreme values.

The Median Percentage Increase was 25 percent with a Predict (20) of 45 percent.⁶³ In Table 10-10, the CV_{est} is 30 percent, and the residuals demonstrate that the Median Percentage neither overestimated nor underestimated a disproportionate amount of the time.

10.4.1.3 MEAN PERCENTAGE POINTS

The mean percent new growth was calculated as follows:

$$\text{Mean Percentage Point Increase} = \frac{\sum (\% \text{Actual New SLOC} - \% \text{Estimated New SLOC})}{n}$$

⁶²When applying the Mean Percentage regression, if the program is greater than 72 percent estimated New SLOC, assume the estimated new SLOC is 100 percent.

⁶³When applying the Median Percentage regression, if the program is greater than 80 percent estimated new SLOC assume the estimated new SLOC is 100 percent.

Section 10 – Risk Analysis

where n is the number of data points in the sample. The resulting mean percentage point estimate is then added directly to the percent estimated new SLOC. The disadvantage was that the mean percentage point estimate was sensitive to extreme values.

| Record # | Mission | Est New | Actual New | A | B | Actual %Inc (B/A) - 1 | C | D | %e(i) (D/B) |
|-------------|---------------|---------|------------|----------|----------|--------------------------|--------------------------|-----------------|----------------|
| | | | | %New Est | %New Act | | Y(est) (0.25 x A) + A | e(i) (C - B) | |
| NCCA-3 | C2 | 191350 | 210550 | 89.50 | 80.42 | -10.14 | 111.87 | 31.4504 | 39.11% |
| NCCA-6 | C2 | 517071 | 519600 | 41.49 | 40.84 | -1.56 | 51.86 | 11.0191 | 26.98% |
| 2461 | Signal Proc | 14800 | 25552 | 95.48 | 96.38 | 0.93 | 119.35 | 22.9795 | 23.84% |
| 2613 | C2 | 90000 | 120000 | 90.00 | 98.36 | 9.29 | 112.50 | 14.1393 | 14.38% |
| 23/24/26/27 | Testing | 36900 | 46303 | 88.28 | 100.00 | 13.28 | 110.35 | 10.3469 | 10.35% |
| 308 | S/W Tools | 20000 | 25000 | 44.44 | 55.56 | 25.00 | 55.56 | 0.0000 | 0.00% |
| NCCA-2 | C2 | 206650 | 394309 | 25.68 | 33.68 | 31.17 | 32.09 | -1.5844 | -4.70% |
| 327 | C2 | 29524 | 119400 | 75.14 | 100.00 | 33.09 | 93.92 | -6.0798 | -6.08% |
| 15/16/17 | C2 | 2763 | 5048 | 11.71 | 19.56 | 67.02 | 14.64 | -4.9201 | -25.16% |
| NCCA-4 | C2 | 65000 | 160000 | 42.48 | 86.49 | 103.58 | 53.10 | -33.3819 | -38.60% |
| 2616 | Mission Plans | 132000 | 537129 | 24.81 | 61.24 | 146.80 | 31.02 | -30.2221 | -49.35% |
| | | | | Mean = | 70.23 | | | | |
| N = 11 | | | | | Median = | 25.00% | SEE = | 21.18 | |
| | | | | | | | CV _{est} = | 0.30 | |
| | | | | | | | Predict (20) = | 45.45% | |

Table 10-10: Code Condition Median Percentage Statistics

As shown in Table 10-11, the mean estimated percentage of new SLOC is 57 percent, while the mean actual percentage is 70 percent, increasing by an average of 13 percentage points.⁶⁴ The Predict (20) is 55 percent, the CV_{est} is 24 percent and the residuals show that the mean percentage point factor overestimated 73 percent of the time.

10.4.1.4 MEDIAN PERCENTAGE POINTS

The median percentage point factor added the “most likely” percentage point directly to the percent estimated new SLOC. It was the central value that divided the data into two groups of equal size and was calculated by taking the difference between percent actual new SLOC and percent estimated new SLOC:

$$\text{Median Percentage Point Increase} = \% \text{Actual New SLOC} - \% \text{Estimated New SLOC}$$

The advantage of this factor was that the median percentage point was relatively insensitive to extreme values, and it improved the Predict (20).

As shown in Table 10-12, the median percentage point Increase is eight percentage points with a Predict (20) of 64 percent⁶⁵ and CV_{est} of 25 percent.

⁶⁴When applying the mean percentage point regression, if the program is greater than 87 percent estimated new SLOC, assume the estimated new SLOC is 100 percent.

⁶⁵When applying the median percentage point regression, if the program is greater than 92 percent estimated new SLOC, assume the estimated new SLOC is 100 percent.

Section 10 – Risk Analysis

| Record # | Mission | Est New | Actual New | A | B | Actual %Pts Inc | C | D | %e(i) |
|-------------|---------------|---------|------------|----------|----------|-----------------|---------------------|----------|---------|
| | | | | %New Est | %New Act | | Y(est) | e(i) | |
| | | | | | | (B - A) | (13 + A) | (C - B) | (D/B) |
| 15/16/17 | C2 | 2763 | 5048 | 11.71 | 19.56 | 7.85 | 24.75 | 5.1988 | 26.59% |
| 23/24/26/27 | Testing | 36900 | 46303 | 88.28 | 100.00 | 11.72 | 101.32 | 1.3235 | 1.32% |
| 308 | S/W Tools | 20000 | 25000 | 44.44 | 55.56 | 11.11 | 57.49 | 1.9349 | 3.48% |
| 327 | C2 | 29524 | 119400 | 75.14 | 100.00 | 24.86 | 88.18 | -11.8179 | -11.82% |
| 2461 | Signal Proc | 14800 | 25552 | 95.48 | 96.38 | 0.89 | 108.53 | 12.1545 | 12.61% |
| 2613 | C2 | 90000 | 120000 | 90.00 | 98.36 | 8.36 | 103.05 | 4.6853 | 4.76% |
| 2616 | Mission Plans | 132000 | 537129 | 24.81 | 61.24 | 36.43 | 37.86 | -23.3792 | -38.18% |
| NCCA-2 | C2 | 206650 | 394309 | 25.68 | 33.68 | 8.00 | 38.72 | 5.0426 | 14.97% |
| NCCA-3 | C2 | 191350 | 210550 | 89.50 | 80.42 | -9.08 | 102.55 | 22.1215 | 27.51% |
| NCCA-4 | C2 | 65000 | 160000 | 42.48 | 86.49 | 44.00 | 55.53 | -30.9569 | -35.79% |
| NCCA-6 | C2 | 517071 | 519600 | 41.49 | 40.84 | -0.65 | 54.54 | 13.6928 | 33.53% |
| N = 11 | | | Mean = | 57.18 | 70.23 | 13 | SEE = | 16.83 | |
| | | | | | | | CV _{est} = | 0.24 | |
| | | | | | | | Predict (20) = | 54.55% | |

Table 10-11: Code Condition Mean Percentage Point Statistics

| Record # | Mission | Est New | Actual New | A | B | Actual %Pts Inc | C | D | %e(i) |
|-------------|---------------|---------|------------|----------|---------------|-----------------|---------------------|----------|---------|
| | | | | %New Est | %New Act | | Y(est) | e(i) | |
| | | | | | | (B - A) | (8 + A) | (C - B) | (D/B) |
| NCCA-3 | C2 | 191350 | 210550 | 89.50 | 80.42 | -9.08 | 97.50 | 17.0755 | 21.23% |
| NCCA-6 | C2 | 517071 | 519600 | 41.49 | 40.84 | -0.65 | 49.49 | 8.6468 | 21.17% |
| 2461 | Signal Proc | 14800 | 25552 | 95.48 | 96.38 | 0.89 | 103.48 | 7.1085 | 7.38% |
| 15/16/17 | C2 | 2763 | 5048 | 11.71 | 19.56 | 7.85 | 19.71 | 0.1528 | 0.78% |
| NCCA-2 | C2 | 206650 | 394309 | 25.68 | 33.68 | 8.00 | 33.68 | -0.0033 | -0.01% |
| 2613 | C2 | 90000 | 120000 | 90.00 | 98.36 | 8.36 | 98.00 | -0.3607 | -0.37% |
| 308 | S/W Tools | 20000 | 25000 | 44.44 | 55.56 | 11.11 | 52.44 | -3.1111 | -5.60% |
| 23/24/26/27 | Testing | 36900 | 46303 | 88.28 | 100.00 | 11.72 | 96.28 | -3.7225 | -3.72% |
| 327 | C2 | 29524 | 119400 | 75.14 | 100.00 | 24.86 | 83.14 | -16.8638 | -16.86% |
| 2616 | Mission Plans | 132000 | 537129 | 24.81 | 61.24 | 36.43 | 32.81 | -28.4251 | -46.42% |
| NCCA-4 | C2 | 65000 | 160000 | 42.48 | 86.49 | 44.00 | 50.48 | -36.0028 | -41.63% |
| | | | | Mean = | 70.23 | | | | |
| N = 11 | | | | | Median %Pts = | 8 | SEE = | 17.73 | |
| | | | | | | | CV _{est} = | 0.25 | |
| | | | | | | | Predict (20) = | 63.64% | |

Table 10-12: Code Condition Median Percentage Point Statistics

10.4.2 APPROACH TWO

The second approach used regression analysis to develop a relationship of the form:

$$\% \text{Actual New SLOC} = a + (b * \% \text{Estimated New SLOC})$$

where:

%Estimated New SLOC = 1 - %Estimated Reused SLOC

Table 10-13 shows the regression results. The R^2 values demonstrated that this approach produced adequate regressions for percent actual new SLOC as presented in the **Approach Three Methodology and Results** section.

| Y(est) | = | X ₁ (coeff) | X ₁ | + | Constant | R ² | CV _{est} | Std ERR | Predict (20) | t(value) | t(sig) | N | df | F(value) | F(sig) |
|--------------|---|------------------------|----------------|---|----------|----------------|-------------------|---------|--------------|----------|--------|----|----|----------|--------|
| %Act New | = | 0.8189 | %Est New | + | 0.234 | 0.7426 | 0.22 | 0.1575 | 45% | 5.0957 | 99.94% | 11 | 9 | 25.9665 | 100% |
| ln(%Act New) | = | 1.4469 | %Est New | + | -1.2919 | 0.6863 | 0.3196 | 0.3196 | 55% | 4.4376 | 99.84% | 11 | 9 | 19.6919 | 100% |
| ln(%Act New) | = | 0.7057 | ln(%Est New) | + | 0.0563 | 0.7965 | 0.2575 | 0.2575 | 55% | 5.9347 | 99.90% | 11 | 9 | 35.2207 | 100% |

Table 10-13: Approach Two - Code Condition Estimating Relationships Results

The best regression equation and its statistics follow:

$$\begin{aligned} \text{\%Actual New SLOC} &= 0.234 + (0.8189 * \text{\%Estimated New SLOC}) \\ R^2 &= 0.74; CV = 0.22; \text{Predict (20)} = 45\%; \text{Range } 16 - 1,246 \text{ KSLOC} \end{aligned} \quad [10-3]$$

Although this approach produced an adequate regression for percent actual new SLOC in the linear form, the Predict (20) was not as good as in Approach One, and the regression tended to overestimate.

10.4.3 RECOMMENDED APPROACH FOR CODE CONDITION CHANGE

Table 10-14 displays a summary of the statistics for the four factors along with the statistics for the most statistically significant regression equation:

| | Estimating Methodology | CV _{est} | Predict (20) |
|----------------|---|-------------------|--------------|
| Mean % | 38% | 0.37 | 36.36% |
| Median % | 25% | 0.30 | 45.45% |
| Mean %Points | 13 | 0.24 | 54.55% |
| Median %Points | 8 | 0.25 | 63.64% |
| Regression Eqn | %Actual New SLOC = 0.234 + (0.82 x %Est New SLOC) | 0.22 | 45.00% |

Table 10-14: Summary of Statistics for Code Condition Methodology

All four factors resulted in low CV_{est}, however both the mean percentage and the mean percentage point factor had disproportionate residuals. To minimize the potential of overestimating a disproportionate amount of the time, NCCA (as stated for the SLOC Growth analysis) prefers to maximize Predict (20) (i.e., be closer to the most likely estimate a higher percentage of the time) rather than minimize overall error. (As stated previously, this is in contrast to our philosophy of minimizing standard error that was applied in earlier sections of this handbook.) Therefore, because the median percentage point factor had the highest Predict (20) of 64 percent and one of the lowest CVs, it is the recommended approach for conducting a top-level risk analysis of the initial code condition assumptions. This approach is applied as follows:

- If estimated new SLOC is greater than or equal to 92 percent, assume the program is 100 percent new SLOC.
- If estimated new SLOC is less than 92 percent, increase the percentage of estimated new SLOC by the median percentage point factor of eight percentage points, and reduce the percentage of reused SLOC by eight percentage points. These changes will, by definition, translate to an increase in the ESLOC count.

10.5 OVERALL RECOMMENDED APPROACH

NCCA's preferred approach to conducting a risk analysis of the initial SLOC estimate requires the analyst to first determine whether contractor or program specific data is available to develop tailored risk analysis relationships. Only if contractor or program specific data is unavailable, should the analyst use these standardized tools to perform the risk analysis. In general, the analyst first increases the estimated total SLOC and then increases the estimated new-to-reused SLOC ratio. Specifically, the analyst should use the following process:

Step 1: Apply the median percent total SLOC growth factor of 22 percent (i.e., as 1.22) to the SLOC estimate. If the SLOC estimate includes reused SLOC, continue to Step 2; otherwise skip to Step 4.

Step 2: If new SLOC is greater than or equal to 92 percent of the total SLOC estimate, assume the program is 100 percent new SLOC and skip to Step 4. If new SLOC is less than 92 percent, proceed to Step 3.

Step 3: Increase the estimated percentage of new SLOC by the median percentage point factor of eight percentage points, and reduce the percentage of reused SLOC by eight percentage points. This changes the ESLOC count. (See the next section for two examples.)

Step 4: Use the output of the steps above as the input to the selected effort estimating methodology and generate an estimate of the associated effort.

Step 5: Estimate the schedule and compute the associated schedule risk by comparing the risk adjusted and non-risk adjusted schedule estimates.

Step 6: Apply the appropriate labor, profit, and G & A rates to the effort estimate.

Step 7: Compute the risk dollars by comparing the risk adjusted and non-risk adjusted (i.e., baseline) cost estimates.

Figure 10-1 below depicts this process:

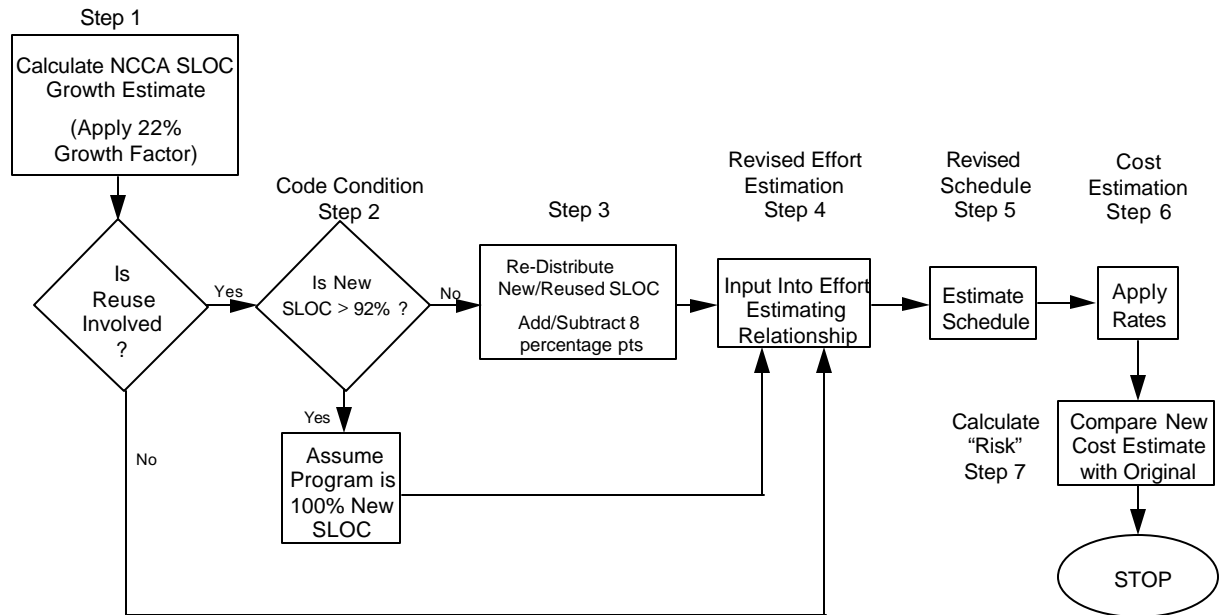


Figure 10-2: Recommended Risk Analysis Process

10.5.1 EXAMPLE ONE

Assumptions for this example are as follows:

Initial SLOC Estimate = 112,450

Initial Code Condition Estimate = 80% New SLOC and 20% Reused SLOC

NCCA Efactor⁶⁶ = 30%

$$\begin{aligned}
 \text{Equivalent New SLOC (1)}^{67} &= \text{New SLOC} + (\text{Reused SLOC} * \text{Efactor}) \\
 &= (0.8 * \text{Total SLOC}) + (0.2 * \text{Total SLOC} * \text{Efactor}) \\
 &= (0.8 * 112,450) + (0.2 * 112,450 * 0.3) \\
 &= 89,960 + 6,747 \\
 &= 96,707
 \end{aligned}$$

Step 1: Assuming contractor/program-specific data is unavailable, the analyst should apply the NCCA standard default factor of a 22 percent increase to the initial estimate of the total SLOC to obtain a revised (i.e., risk) total SLOC count.

$$\begin{aligned}
 \text{Revised Total SLOC} &= (0.22 * \text{Initial SLOC Estimate}) + \text{Initial SLOC Estimate} \\
 &= (0.22 * 112,450) + 112,450 \\
 &= 137,189
 \end{aligned}$$

⁶⁶Code condition will have been provided, and the analyst will have already solved for the Efactor in effort estimation.

⁶⁷For example purposes only. In practice, if the analyst is using NCCA standard or tailored regressions, the ESLOC calculation would occur in the effort estimation procedure.

This increase in total SLOC count translates to an increase in the ESLOC count as follows:

$$\begin{aligned}\text{Revised Equivalent} &= \text{New SLOC} + (\text{Reused SLOC} * \text{Efactor}) \\ \text{New SLOC} &= (0.8 * \text{Revised Total SLOC}) + (0.2 * \text{Revised Total SLOC} * 0.3) \\ \text{(Code growth only)} &= (0.8 * 137,189) + (0.2 * 137,189 * 0.3) \\ &= 109,751 + 8,231 \\ &= 117,982\end{aligned}$$

Step 2: Determine whether the %new code is greater than 92 percent. Since it is not (80 percent) proceed to Step 3.

Step 3: To obtain the revised new SLOC versus reused SLOC counts, the analyst should increase the new SLOC percentage by eight percentage points and decrease the reused SLOC percentage by eight percentage points:

$$\begin{aligned}\text{Estimated New SLOC with Growth:} \\ &= 0.80 \text{ New} * \text{Revised Total SLOC} \\ &= 0.80 * 137,189 \\ &= 109,751\end{aligned}$$

Estimated New SLOC with Growth and code condition adjustment:

$$\begin{aligned}&= [.80 (\% \text{New}) + 0.08 (\text{Code Condition Adj})] * \text{Revised Total SLOC} \\ &= 0.88 * 137,189 \\ &= 120,726\end{aligned}$$

and

Estimated Reused SLOC with growth:

$$\begin{aligned}&= 0.20 (\text{Reused SLOC}) * \text{Revised Total SLOC} \\ &= 0.20 * 137,189 \\ &= 27,438\end{aligned}$$

Estimated Reused SLOC with growth and code condition adj:

$$\begin{aligned}&= [0.20 (\text{Reused}) + 0.08 (\text{Code Condition Adj})] * \text{Revised Total SLOC} \\ &= 0.12 * 137,189 \\ &= 16,463\end{aligned}$$

or alternatively,

Estimated Reused SLOC with growth and code condition adj:

$$\begin{aligned}&= \text{Revised Total SLOC} - \text{Est New SLOC with Growth \& Code Condition Adj} \\ &= 137,189 - 120,726 \\ &= 16,463\end{aligned}$$

These revised estimates of new SLOC versus reused SLOC translate into a revised ESLOC count.

$$\begin{aligned}
 \text{Equivalent New SLOC}^{68} &= (\text{Revised New SLOC}) + (\text{Revised Reused SLOC} * \text{Efactor}) \\
 \text{(Code growth and code condition)} &= (120,726) + (16,463 * 0.3) \\
 &= 120,726 + 4,939 \\
 &= 125,665
 \end{aligned}$$

Step 4: Input the revised ESLOC into the effort estimation process.

Step 5: Estimate the schedule and compute the associated schedule risk by comparing the risk adjusted and non-risk adjusted (i.e., baseline) schedule estimates.

Step 6: Apply the appropriate labor, profit, and G & A rates.

Step 7: Compute the risk dollars by comparing the risk adjusted and non-risk adjusted (i.e., baseline) cost estimates.

In summary, total code growth and code condition risk is 28,958 ESLOC, or the difference between 96,707 and 125,665. This difference represents approximately a 30 percent increase in the ESLOC count.

10.5.2 EXAMPLE TWO

Assumptions for this example are as follows:

$$\begin{aligned}
 \text{Initial SLOC Estimate} &= 112,450 \\
 \text{Initial Code Condition Estimate} &= 94\% \text{ New SLOC and } 6\% \text{ Reused SLOC} \\
 \text{NCCA Efactor} &= 30\% \\
 \\
 \text{Equivalent New SLOC} &= \text{New SLOC} + (\text{Reused SLOC} * \text{Efactor}) \\
 &= (0.94 * \text{Total SLOC}) + (0.06 * \text{Total SLOC} * \text{Efactor}) \\
 &= (0.94 * 112,450) + (0.06 * 112,450 * 0.3) \\
 &= 105,703 + 2,024 \\
 &= 107,727
 \end{aligned}$$

Step 1: Assuming contractor/program specific data is unavailable, the analyst should apply the NCCA standard default factor of a 22 percent increase to the initial estimate of the total SLOC to obtain a revised (i.e., risk) total SLOC count:

$$\begin{aligned}
 \text{Estimated Total SLOC with Growth:} &= (0.22 * \text{Initial SLOC Estimate}) + \text{Initial SLOC Estimate} \\
 &= (0.22 * 112,450) + 112,450 \\
 &= 137,189
 \end{aligned}$$

Step 2: In this case, new SLOC is greater than 92 percent; therefore, new SLOC increases to 100 percent with no reused SLOC; that is, 137,189 is the total amount of SLOC, and it is all new SLOC (i. e., total SLOC equals ESLOC).

⁶⁸For example purposes only. In practice, if the analyst is using NCCA standard or tailored regressions, the ESLOC calculation would occur in the effort estimation procedure.

Step 3: Skip, since the SLOC are all new.

Step 4: Input the revised ESLOC (also equal to total SLOC) into the effort estimation process.

Step 5: Estimate the schedule and compute the associated schedule risk by comparing the risk adjusted and non-risk adjusted (i.e., baseline) schedule estimates.

Step 6: Apply the appropriate labor, profit, and G & A rates.

Step 7: Compute the risk dollars by comparing the risk adjusted and non-risk adjusted (i.e., baseline) cost estimates.

In summary, total code growth and code condition risk is 29,462 ESLOC, or the difference between 107,727 and 137,189. This difference represents approximately a 27 percent increase in the ESLOC count.

10.6 FUTURE EFFORTS

Future efforts to improve the SLOC Growth Analysis should consider the following:

- 1) Expanding the NCCA Normalized SLOC Growth Database to include more program-level data points, particularly programs greater than 100 KSLOC. This should result in a database that is more normally distributed than the current database, which would decrease the uncertainty of estimating future programs where the size lies outside the current range.
- 2) Expanding the NCCA Normalized SLOC Growth Database to include MIS data points. Since the development of MIS programs is increasing, NCCA needs insight into how and why these programs grow. Additionally, since it is generally agreed that function point estimating is more appropriate for MIS programs, differences in growth may be experienced.
- 3) Researching additional explanatory variables to gain insight into SLOC growth. This analysis failed to uncover the reasons for SLOC code growth (i. e., requirements creep, poor estimating, etc.).
- 4) Identifying the timing of the initial estimates by phase (SDR, SSR, FQT, etc.) for the current NCCA Normalized SLOC Growth Database and any new data points to determine when SLOC growth occurs. This would show when growth is most likely to occur, thus facilitating the development of appropriate funding profiles.
- 5) Identifying the impact of schedule on SLOC growth. This is important because increases or decreases in schedule may lead to adjustments in requirements which may then lead to corresponding changes in size.

CONCLUSIONS

Because software development is influenced by so many factors which either are not or can not be captured quantitatively, software cost estimating will remain a great challenge.

NCCA has attempted to remove some of the subjectivity involved in software cost estimating by developing effort, schedule, labor rate and risk estimating relationships and factors based primarily on objective or easily quantifiable parameters. However, because the regressions and factors are top-level and reflect industry averages, the resulting standard errors for effort and schedule are typically above 40 percent. Although the resulting statistics for the regressions and factors of the labor rate and risk analyses were much better in comparison, the limited size of the underlying databases causes concern.

Many of the effects NCCA tried to measure (such as the effect of embedded versus non-embedded development) are difficult to isolate. These analyses were not performed with data from an experimental environment in which factors could be controlled. Therefore, in an attempt to control for these factors, the data was filtered into specific subsets. Clearly, analysts would have to start with a huge database to isolate the effect of more than a few specific factors.

Analysts need to pay special attention to the fact that the NCCA standard effort regressions represent industry averages. They represent both old and new processes and tools, well-behaved and ill-behaved programs, and relatively simple and complex programs. Therefore, these top-level regressions should **NOT** be the estimating tools of choice. Instead, contractor-specific data should be collected for the projects being estimated, where the data represents **completed** projects that are analogous to both the project being estimated and the type of process that will be used to develop it. If, due to lack of data, analysts choose to utilize these tools, they should realize and explicitly state the associated variance. In fact, if possible, the statistical range resulting from the tools, in lieu of, or at least in addition to, the point estimate, should be provided.

To truly capture the significant software development drivers while decreasing the associated variances, the data utilized to develop software development regressions needs to be:

- From the same contractor
- From the same software development staff
- For a similar set of requirements
- From a similar set of tools and processes
- From a similar mission environment
- At the same level of complexity

Section 11 – Conclusions

Since this would require an extensive number of programs as well as effort and time, the best alternative in the interim is to develop the most analogous set of normalized data available.

Collecting more data, sensitive to the list provided above, would lead to better overall top-level regressions. However, it is unlikely that the variances of top-level regressions will ever approach the variances of lower-level, domain, mission and contractor-specific regressions. Furthermore, in order to ensure the integrity of lower-level regressions, a sufficient quantity of data points must be obtained.

In conclusion, the Navy's ability to estimate software development cost is directly related to the quantity and quality of data collected for completed efforts. Because there are so many variables affecting software development productivity, cost, and schedule, a concerted data collection effort is required to improve our estimating tools. By collecting all the data currently available to NCCA and creating normalized databases, NCCA has taken the first and most critical step in a challenging process of software cost estimating. The next steps will focus on those areas of the database which are deficient, in an attempt to further capture objective, significant software productivity drivers and to increase the associated scope to which the tools can reasonably be applied. It is our hope that, at a minimum, the analyst is now aware of those variables which should be considered and addressed when gathering data and developing software cost estimates.

REFERENCES

- [1] Frazier, Thomas, Bailey, John and Young, Melissa, "*Comparing ADA and Fortran LOC: Some Experimental Results*," IDA Paper P-2899, November 1993.
- [2] Software Engineering Laboratory, "*Impact of Ada and Object-Oriented Design in the Goddard Space Flight Center*", SEL-95-001, March 1995.
- [3] Parks, Robert, "*Software Size Measurement: A Framework for Counting Source Statements*," Software Engineering Institute, CMU/SEI-92-TR-20, 1992.
- [4] Frieden, David R., Principles of Naval Weapons Systems, Naval Institute Press, 1985.
- [5] Boehm, Barry, Software Engineering Economics, Prentice Hall, 1981.
- [6] Ratliff, Robert W., Bowden, Robin G., and Cheadle, William, "*SASET 3.0 User's Guide*," April 1993.
- [7] Novak-Ley, Gina and Stukes, Sherry, "*Space and Missile Systems Center Software Database, User's Manual*" Version 1.0, MCR.
- [8] Funch, Paul, "*Software Cost Data Base*," MITRE Study #MTR 10329, October 1987.
- [9] Giallombardo, Robert J., "*Effort and Schedule Estimating Models for Ada Software Developments*", MITRE Corporation, #MTR11303, May 1992.
- [10] Software Engineering Laboratory, "*Cost and Schedule Estimation Study Report*", SEL-93-002, November 1993.
- [11] Software Engineering Laboratory, "*Recommended Approach to Software Development, Revision 3*," SEL-81-305, June 1992.
- [12] Dechoretz, Jason and Stukes, Sherry, "*Software Estimating Model Improvement Program REVIC Recalibration*," MCR Study #TR-9359-51-B, April 1994.
- [13] IITRI Research Institute, "*Test Case Study: Estimating the Cost of Ada Software Development*," IITRI Report , April 1989.
- [14] Spatz, Chris and Johnston, James O., Basic Statistics, 3rd Edition, Brooks/Cole Publishing Company, 1984.
- [15] Software Technology Support Center, Department of the Air Force, *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems, command and control Systems, Management Information Systems*, Vol 1, Version 1.1, U.S. Government Printing Office, Washington, D.C., Feb 1995

[16] Herd, James, et al, "*Software Cost Estimation Study: Study Results*," Doty Associates, RADC-TR-77-220, Volume 1, June 1977.

[17] Wolverton, R.W., "*Software Costing*," Handbook of Software Engineering, Van Nostrand Reinhold Company, pp 469-493, 1984.

[18] Pressman, Roger S., Software Engineering A Practitioner's Approach, 3rd Edition, McGraw Hill, 1992.

[19] Hines, William W. and Montgomery, Douglas C., Probability and Statistics in Engineering and Management Science, Second Edition, John Wiley and Sons, 1980.

[20] J. Verner and G. Tate, "A Software Size Model," *IEEE Transactions on Software Engineering*, Vol. 18, No. 4, pp. 265-278, April, 1992.

[21] S. D. Conte, Dunsmore, H. E., and Shen, V. Y., Software Engineering Metrics and Models, Menlo Park, CA: Benjamin/Cummings, 1986.

[22] Om, N. and Bui, J., "Software Development Cost and Schedule Estimating for Space Systems," Report for the Ballistic Missile Defense Organization, June 1994.

ACRONYM LIST

| | |
|----------------|---|
| 498 | MIL-STD-498 |
| 2GL | Second-Generation Language |
| 3GL | Third-Generation Language |
| 4GL | Fourth-Generation Language |
| AAF | Adaptation Adjustment Factor |
| ACO | Administrative Contracting Officer |
| ACWP | Actual Cost of Work Performed |
| AIS | Automated Information System |
| ASW | Anti-Surface Warfare |
| C ² | Command and Control |
| C ³ | Command, Control, and Communications |
| CCDR | Contractor Cost Data Report |
| CDR | Critical Design Review |
| CDRL | Contract Data Requirements List |
| CED | Concept Exploration and Definition |
| COCOMO | Constructive Cost Model |
| COTS | Commercial-Off-The-Shelf |
| CP | Commented Physical |
| CPCI | Computer Program Configuration Item |
| CPR | Cost Performance Report |
| CSC | Computer Software Component |
| CSCI | Computer Software Configuration Item |
| CSU | Computer Software Unit |
| CV | Coefficient of Variation |
| DCAA | Defense Contracting Audit Agency |
| DEM/VAL | Demonstration and Validation |
| DoD | Department of Defense |
| DOS | Disk Operating System |
| DSI | Delivered Source Instructions |
| DSLOC | Delivered Source Lines Of Code |
| EAF | Effort Adjustment Factor |
| Efactor | Equivalent Code Conversion Factor |
| EMD | Engineering and Manufacturing Development |
| ESD | Electronic Systems Division |
| ESLOC | Equivalent New Source lines of code |
| EW | <i>Electronic Warfare</i> |
| FPRA | Forward Pricing Rate Agreement |
| FQT | Formal Qualification Test |
| FW | Firmware |

| | |
|---------|--|
| G&A | General and Administrative |
| HOL | High Order Language |
| ICE | Independent Cost Estimate |
| IDA | Institute for Defense Analyses |
| IOC | Initial Operational Capability |
| IITRI | IIT Research Institute |
| KSLOC | Thousands of Source Lines of Code |
| L | Logical |
| LRE | Latest Revised Estimate |
| MAD | Mean Absolute Deviation |
| MCCR | Mission Critical Computer Resources |
| MCR | Management Consulting and Research, Inc. |
| MIL-STD | Military Standard |
| MIS | Management Information System |
| MM | Man-Months |
| NCCA | Naval Center for Cost Analysis |
| OFP | Operational Flight Program |
| OTE | Operational Test and Evaluation |
| P | Physical |
| PDL | Program Design Language |
| PDR | Preliminary Design Review |
| PDRR | Program Definition and Risk Reduction |
| RC | %Re-Code |
| RD | %Re-Design |
| REVIC | Revised Intermediate COCOMO |
| RFP | Request for Proposal |
| RT | %Re-Test |
| SASET | Software Architecture Sizing & Estimating Tool |
| SDP | Software Development Plan |
| SDR | System Design Review |
| SEE | Standard Error of the Estimate |
| SEL | Software Engineering Laboratory |
| SIT | System Integration and Test |
| SLOC | Source Lines of Code |
| SMC | Space and Missile Center |
| SRS | Software Requirements Specification |
| SSCAG | Space Systems Cost Analysis Group |
| SSR | Software Specification Review |
| STP | Software Test Plan |
| WBS | Work Breakdown Structure |